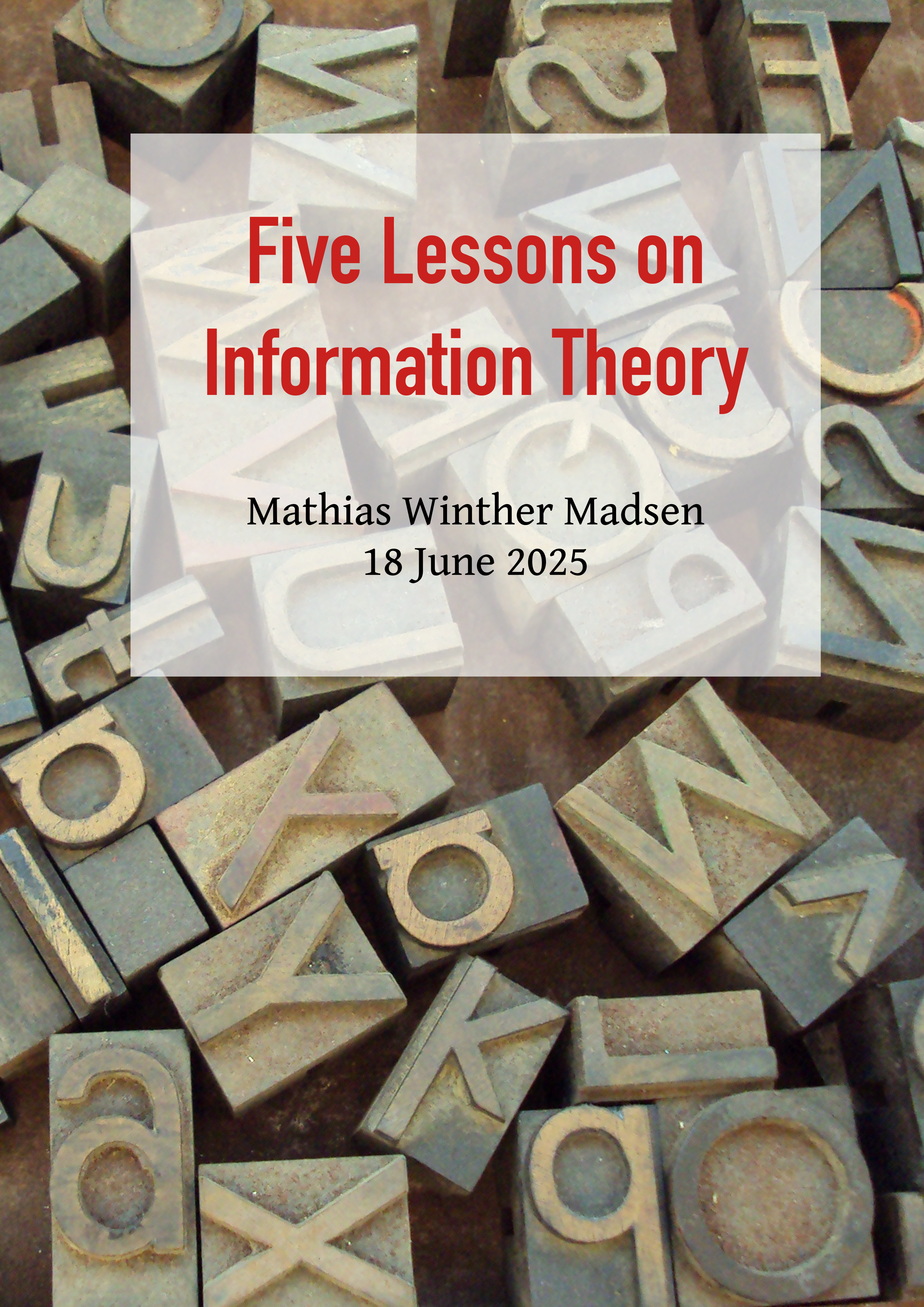


Five Lessons on Information Theory

Mathias Winther Madsen

18 June 2025

Five Lessons on Information Theory

Mathias Winther Madsen

mathias.winther@gmail.com

June 18, 2025

This book was prepared for the 2025 North American Summer School in Logic, Language, and Information in Seattle, Washington.

References in the exercises are to MacKay: *Information Theory, Inference and Learning Algorithms* (2003) and Cover and Thomas: *Elements of Information Theory* (1991).

Contents

Chapter 1

MONDAY

1.1 Introduction

Information theory is a branch of probability theory that deals with questions concerning the encoding and decoding of messages for purpoees of data compression, noise reduction, or secrecy.

The field can with some justice be said to have been invented by Claude Shannon in a 1948 paper published when he was a research engineer for the Bell Telephone Company.[1] That paper proves a theoretical bound on lossless data compression and a constructive method for approximating that bound, and proves that it is possible achieve nearly noise-free communication is possible over noisy communication channel with a clever choice of communication protocol. These results are the theoretical foundation behind many practical techniques used everywhere in digital communication today, and they have interesting consequences for linguistics, epistemology, and other fields.

The majority of this course is dedicated to problems surrounding data compression, but we will also look at various other topics in information theory. On the last day of the course, we will prove Shannon's channel coding theorem about the limits of reliable communication in the presence of noise.

1.2 Redundancy

Exercise 1. Decipher as much as possible of the following text:

```
THE CHIEF DIFFICULTY ALICE FOUND AT [redacted]IRST WAS IN MANAG[redacted]NG
[redacted]ER FLA[redacted]INGO: SHE SUCCEEDED IN GET[redacted]ING I[redacted] BOD[redacted] TUCKED
[redacted]WAY, C[redacted]MFORTABLY ENOU[redacted], UN[redacted]E[redacted] HER AR[redacted], WITH IT[redacted] LEGS
HANG[redacted]NG [redacted]OWN, BU[redacted] GENE[redacted]A[redacted]L[redacted], [redacted]U[redacted]T [redacted] S[redacted]E H[redacted] [redacted]OT [redacted]S
[redacted]CK NI[redacted]ELY [redacted]T[redacted]AI[redacted]NE[redacted] [redacted]UT, AND [redacted]AS G[redacted]ING T[redacted] G[redacted]VE
THE HEDGEHO[redacted] [redacted] B[redacted] [redacted]I[redacted] I[redacted] H[redacted]D, [redacted]T W[redacted]U[redacted]D [redacted]W[redacted]ST
I[redacted]SEL[redacted] R[redacted]UND [redacted]D L[redacted] [redacted] [redacted]N H[redacted]R [redacted]C[redacted], WIT[redacted] S[redacted]CH [redacted]
[redacted]Z[redacted]E[redacted] [redacted]PR[redacted]N [redacted]T [redacted] [redacted]O[redacted]D NO[redacted] [redacted]LP
[redacted]U[redacted]T[redacted] O[redacted] [redacted]GH[redacted]: [redacted] [redacted]N [redacted]E HA[redacted] [redacted]T [redacted]
```

[1]Claude Shannon: "A Mathematical Theory of Communication," *Bell System Technical Journal*, 1948.

[redacted]E[redacted] [redacted]W[redacted], [redacted]ND [redacted] [redacted]I[redacted] [redacted] [redacted]E[redacted]N [redacted]N, IT [redacted]
[redacted]R[redacted] [redacted]R[redacted]I[redacted] [redacted] [redacted]N[redacted] T[redacted] T[redacted] [redacted]EDG[redacted]O[redacted] [redacted]
[redacted]L[redacted]D [redacted]SE[redacted], [redacted]D [redacted]A[redacted] [redacted] [redacted] [redacted] [redacted] [redacted]R[redacted]W[redacted]
[redacted].

This text was prepared by randomly blacking out a larger and larger proportion of the letters in the text. A experience when solving this exericse is that the text can be reconstructed perfectly in the beginning, but at some point in the middle, the noise becomes too overwhelming to guess with any confidence what the blacked-out letters were.

This phenomenon is due to the **redundancy** of the English language: not every letter is strictly necessary for the message to be carried across, making the language robust to a certain amount of corruption. This observation raises two important possibilities:

1. we could potentially achieve a better **data compression** by removing redundancy, i.e., not transmitting the letters that weren't strictly necessary, in order to save transmission energy or storage space

2. but conversely, we could also search more systematically for ways of adding redundancy to a language in a way that would make it more robust to noise, effectively turning it into an **error-correcting code**.

We will spend the greater part of the course studying data compression, and then turn to error-correcting codes.

1.3 Entropy

A key statistic which will play a role in data compression is **entropy**. It is a property of a probability distribution, which for our purposes can be throught of as a discrete set of **probability masses** $p(x_1), p(x_2), p(x_3), \ldots$. The entropy of such a probability distribution is defined as

$$H = \sum_x p(x) \log_2 \frac{1}{p(x)},$$

where $\log_2$ is the binary logarithm.

Exercise 2. Find the entropy of the probability distributions with the following sets of probability masses:

1.

x	A	B
$p(x)$	1/2	1/2

2.

x	A	B	C	D
$p(x)$	1/2	1/4	1/8	1/8

3.

x	A	B	C
$p(x)$	1/3	1/3	1/3

4.

x	A	B	C
$p(x)$	1/2	1/3	1/6

Solution. Here are the solutions to the first two exercises:

1.

$$\frac{1}{2}\log_2\frac{2}{1}+\frac{1}{2}\log_2\frac{2}{1} = \frac{1}{2}+\frac{1}{2} = 1.$$

2.

$$\frac{1}{2}\log_2\frac{2}{1}+\frac{1}{4}\log_2\frac{4}{1}+\frac{1}{8}\log_2\frac{8}{1}+\frac{1}{8}\log_2\frac{8}{1} = \frac{1}{2}+\frac{2}{4}+\frac{3}{8}+\frac{3}{8} = 1\frac{3}{4}.$$

You can use the approximation $\log_2 3 \approx 1.58$ for the last two.

Entropy is a measure of randomness, or uncertainty. Distributions over many equally likely outcomes have high entropy, whereas distributions that are heavily skewed towards a small set of outcomes have a low entropy.

In the case of a distribution with two possibly outcomes, the entropy is 1 when when the two outcomes are equally likely, and 0 when the one of the two outcomes happens with probability 1. The entropy of such a biased coin flip is called the **binary entropy**.

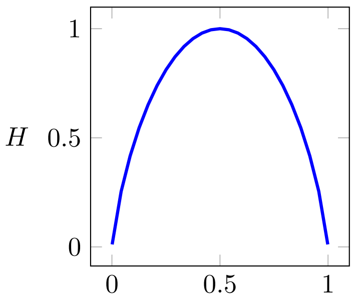


Figure 1.3.1: Binary entropy.

The entropy of a random experiment can be thought of telling us how many flips of a fair coin the experiment corresponds to. For instance, rolling a single die creates the same amount of uncertainty as $\log_2 6 \approx 2.58$ fair coin flips.

We say that entropy is measured in binary digits, or **bits**. If we used logarithm with a base of 10 instead of a base of 2, the unit would instead be **digits**. The entropy of a distribution would then measure its uncertainty as a number of choices of digits from $\{0, 1, 2, \ldots, 9\}$.

The significance of the concept of entropy is, as we will see later, that it quantifies the limit on how much a source of information can be compressed. The results of a series of independent experiments with entropy H can be communicated at an average cost close to H binary digits per experiment, and not lower.

1.4 Unique Decodability

The simplest way of encoding a letter from a given alphabet as a sequence of binary digits is to pick an equally long codeword for each letter:

A	B	C	$\cdots$	Z
00000	00001	00010		11001

For an alphabet of k messages, this requires $\log_2 k$ binary digits per letter, rounded up to the nearest integer.

This coding scheme is easy to encode and decode, but achieves no compression. To encode the alphabet in a way that achieves a lower average message length, we can leverage the fact that some letters are more probable than others.

However, we have to be a little bit careful about how we choose such sets of codewords to avoid creating ambiguities for the decoder. A common but wasteful solution is to set aside a special "space character" that signals the end of each codeword, so that it's clear how to parse a received message into individual codewords. This is however not the only solution, and the general aim is the following:

Definition 3. A set of codewords is uniquely decodable if no two sequences of codewords produce identical texts.

Exercise 4. Which of the following sets of codewords are uniquely decodable?

1. $\{\texttt{00}, \texttt{0}, \texttt{1}\}$
2. $\{\texttt{0}, \texttt{10}, \texttt{11}\}$
3. $\{\texttt{0}, \texttt{01}, \texttt{11}\}$
4. $\{\texttt{01}, \texttt{10}, \texttt{1001}\}$
5. $\{\texttt{00}, \texttt{1100}, \texttt{1111}\}$

Solution.

1. Not uniquely decodable, since the text `00` could be the single word `00` or the two words $\texttt{0} + \texttt{0}$.
2. Uniquely decodable; we can parse the received text from left to right, popping off one binary digit whenever the next symbol is `0`, and two binary digits when the next symbol is `1`.
3. Also uniquely decodable; this is the same code as above, flipped left to right, so we can use the same parsing strategy, only consuming the string from left to right.
4. Note uniquely decodable, since the codeword `1001` is the concatenation of the two other codewords.
5. Uniquely decodable, since it is the same as the second code, but in blocks of two identical bits.

Although I will not cover the details here, it is worth noting that an algorithm exists for determining whether a given code is uniquely decodable or not.[2]

Definition 5. A **prefix code** is a set of codewords with the property that no codeword is a prefix (initial segment) of any other.

[2] Sardinas and Patterson: "A Necessary and Sufficient Condition for Unique Decomposition of Coded Messages," *IRE Convention Record*, 1953.

Of the examples above, $\{\mathtt{0}, \mathtt{10}, \mathtt{11}\}$ is a prefix code, and $\{\mathtt{0}, \mathtt{01}, \mathtt{11}\}$ is not.

It is often useful to think of a binary string as a sequence of left/right decisions that describes a path through a binary tree. A code is then a set of nodes in this tree; a prefix code is a set of nodes with the property that the path from the root to one node in the set never passes another node in the set. In other words, a prefix code is a set of leaves in a binary tree.

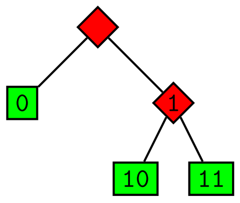


Figure 1.4.1: A prefix code represented as a tree.

Such a code tree also conveniently doubles of a state diagram for a program that can be used to parse an incoming bitstream: every incoming bit represents a step down the binary tree, either left or right, and every time we step onto a node that represents a codeword, we output the corresponding codeword and return to the root.

This observation essentially proves the following theorem:

Theorem 6. *All prefix codes are uniquely decodable.*

The converse of this theorem is not quite true. A slightly weaker theorem does hold, however, as we will see below.

1.5 The Kraft-McMillan Theorems

As we have already seen, there are uniquely decodable codes which are not prefix codes. The following theorem states that we can always replace such non-prefix codes with an equally good prefix codes:

Theorem 7. *For any uniquely decodable code, there is a prefix code with the same codeword lengths.*

For example, the code $\{\mathtt{0}, \mathtt{01}, \mathtt{11}\}$ is uniquely decodable but not a prefix code; however, the code $\{\mathtt{0}, \mathtt{10}, \mathtt{11}\}$ has the same codeword lengths and is a prefix code.

This theorem can be proven in two steps:

Theorem 8 (McMillan's Theorem)**.**[3] *If a code is uniquely decodable, then the codeword lengths $w_1, w_2, \dots, w_k$ satisfy the inequality*

$$\sum_{i=1}^{k} 2^{-w_i} \;\leq\; 1.$$

If we say that the **footprint** of a codeword of length w is 2^{-w}, then McMillan's theorem tells us that a code cannot be uniquely decodable if its total footprint is too large.

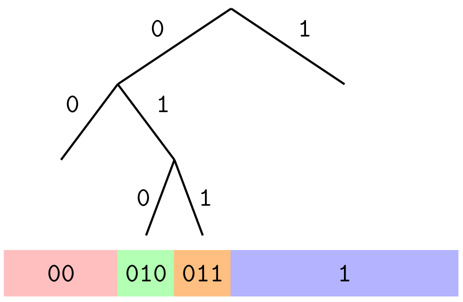
[3]Brockway McMillan: "Two Inequalities Implied by Unique Decipherability," *IEEE Transactions on Information Theory*, 1956.

Theorem 9 (Kraft's Inequality).[4] *If a set of integers $w_1, w_2, \dots, w_k$ satisfy the inequality*

$$\sum_{i=1}^{k} 2^{-w_i} \leq 1,$$

then there is a prefix code with codeword lengths $w_1, w_2, \dots, w_k$.

These two theorems highlight an important correspondence between prefix codes, binary trees, and subdivisions of the unit interval: picking a codeword of length w corresponds to picking an interval $[a, b) \subseteq [0, 1)$ whose width is a power of 2, and whose left-hand boundary is an integer multiple of its width.



Proof of Kraft's inequality. Since the integers $w_1, \dots, w_k$ satisfy Kraft's inequality, we can clearly choose intervals of length $2^{-w_1}, \dots, 2^{-w_k}$ that fit inside of the unit interval. However, we need to pick intervals whose left-hand boundaries are located at integer multiples of their widths. We can accomplish this by eating away at the unit interval from the left, starting with the largest bites. Then the left-hand boundary of the remaining part of the interval will always be an integer multiple of the width of the next bite, since the the widths decrease by positive integer factors with each step. □

The method described in this proof is equivalent to sorting the codeword lengths $w_1, \dots, w_k$ and choose codewords lexicographically—for instance, choose the codewords (`0`, `10`, `110`, `111`) if the codeword lengths are $(1, 2, 3, 3)$. It always results in a right-branching code tree in which all right-hand subtrees have larger or equal depth to their left-hand siblings. These are not the only valid code trees consistent with a set of codeword lengths, but the theorem only requires us to exhibit one possible solution.

In order to prove McMillan's theorem, we will have to clarify the concept of a **sentence**, which is a sequence of codewords. The **yield** of a sentence is the binary string that results from concatenating the words in the sentence. The sentence (`10`, `0`, `11`) thus yields the string `10111`. A code is uniquely decodable if and only if there are no two sentences with identical yields.

The length of a sentence is the sum of its codeword lengths. Since

$$2^{-(w_1+w_2+\cdots+w_n)} = 2^{-w_1}2^{-w_2}\cdots 2^{-w_n}$$

the footprint of a sentence is the product of the codeword footprints. This suggests an interesting proof of McMillan's theorem, originally due to Doob.

Proof of McMillan's Theorem. If a code is uniquely decodable, then it cannot produce more than 2^b sentences with a yield of b bits, since there are only 2^b

[4]Leon G. Kraft: *A Device for Quantizing, Grouping, and Coding Amplitude Modulated Pulses*, MIT master's thesis, 1949.

binary strings of length b. The footprint of the set of sentences with a yield of b bits can therefore be no larger than $2^b 2^{-b} = 1$ if the code is uniquely decodable.

Let w^* be an upper bound on the codeword lengths, $w_1, \ldots, w_k \leq w^*$. Then the yield of any sentence with n words has some length between 0 and nw^*. Every sentence with n or fewer words can therefore be put into one of nw^* bins according to the length of its yield. The total footprint of all sentences with n or fewer words is therefore no more than

$$\underbrace{1 + 1 + \cdots + 1}_{\text{lengths } 0, 1, \ldots, nw^*} \;=\; nw^* + 1.$$

Note that this is a linear function of n.

On the other hand, the exponential function

$$\left(\sum_{i=1}^{k} 2^{-w_i} \right)^n \;=\; \sum_{i_1=1}^{k} \sum_{i_2=1}^{k} \cdots \sum_{i_n=1}^{k} 2^{-(w_{i_1} + w_{i_2} + \cdots + w_{i_n})}$$

contains every possible combination of n codeword lengths, and therefore sums up the footprint of every possible sentences with exactly n words. If

$$\sum_{i=1}^{k} 2^{-w_i} \;>\; 1,$$

then this total footprint is an exponentially increasing function in n. Since an exponentially increasing function in n cannot be smaller than a linear function in n for every n, the code cannot both be uniquely decodable and have a footprint larger than 1. □

1.6 A Lower Bound on Compression

The Kraft-McMillan theorems show that uniquely decodable codes with codeword lengths $w_1, \ldots, w_k$ exist if and only if

$$\sum_{i=1}^{k} 2^{-w_i} \;\leq\; 1.$$

Assuming the letters of a given alphabet occur with probabilities $p_1, \ldots, p_k$, a code with these particular codeword lengths will have an average codeword length of

$$p_1 w_1 + p_2 w_2 + \cdots + p_k w_k.$$

The problem of minimizing average codeword length subject to unique decodability can therefore be formulated as the constrained minimization problem

$$\begin{aligned} \underset{w_1, \ldots, w_k}{\text{minimize}} & \sum_{i=1}^{k} p_i w_i \\ \text{subject to} & \sum_{i=1}^{k} 2^{-w_i} \leq 1 \end{aligned}$$

where the numbers $w_1, \dots, w_k$ must be integers.

We can compute a lower bound on the solution of this problem by allowing not only integer vectors, but any nonnegative, real-valued vectors. Without changing the value of the objective function, we can also make the substitution $q_i = 2^{-w_i}$ (that is, $w_i = -\log_2 q_i$). The relaxed problem then takes the form

$$\begin{aligned} \underset{q_1,\dots,q_k}{\text{minimize}} \sum_{i=1}^{k} p_i \log_2 \frac{1}{q_i} \\ \text{subject to} \sum_{i=1}^{k} q_i \leq 1 \end{aligned}$$

where $q_1, \dots, q_k$ are nonnegative, real-valued numbers.

The solution to this minimization problem is $q_i = p_i$ for all i. At this solution, the value of the objetive function is

$$\sum_{i=1}^{k} p_i \log_2 \frac{1}{q_i} = \sum_{i=1}^{k} p_i \log_2 \frac{1}{p_i},$$

which is the entropy of the distribution with probability masses $p_1, \dots, p_k$. The entropy is thus a lower bound on the average codeword length of all uniquely decodable codes. In the following sections, we will exhibit some practical codes that approximate this lower bound from above.

1.7 Shannon Coding

We will now introduce a method of coding that approximates the Shannon compression limit. It was described by Shannon himself in his 1948 paper.[5]

Exercise 10. We need to inspect 2 digits to distinguish the decimal fractions

$$\begin{aligned} c_1 &= .7071067811\dots \\ c_2 &= .7724538509\dots \end{aligned}$$

In general, how many digits do we need to inspect in order to distinguish two given decimal fractions $0 \leq c_1 < c_2 < 1$?

The solution to this exercise suggests an idea for a coding scheme for an alphabet sorted such that the letter probabilities $p(x_1), p(x_2), p(x_3), \dots, p(x_k)$ are in decreasing order. The coding scheme consists in approximating the cumulative probabilities with a decimal fraction just long enough to make sure that we get a prefix code:

[5]Claude Shannon: "A Mathematical Theory of Communication," *Bell System Technical Journal*, 1948.

x	point mass	cumulative	$-\log_{10} p(x)$	w	codeword
A	.41131009	.00000000	0.38583064	1	0
B	.35430328	.41131009	0.45062483	1	4
C	.16862789	.76561337	0.77307059	1	7
D	.03420683	.93424126	1.46588717	2	93
E	.02861911	.96844809	1.54334388	2	96
F	.00176739	.99706720	2.75266761	3	997
G	.00098308	.99883459	3.00741114	4	9988
H	.00018233	.99981767	3.73914187	4	9998

We pick the number of digits such that the approximation of c_i differs from the approximation of c_{i+1}. Since the letter probabilities are sorted, this also ensures that it differs from c_{i-1}.

The use of decimal fractions here is purely illustrative, and we can use the same scheme with binary fractions:

x	$p(x)$	$\Sigma\, p(x)$	$\Sigma\, p(x)$ in binary	$-\log_2 p(x)$	w	codeword
A	.350	.00	.00000000...	1.515	2	00
B	.250	.35	.01011001...	2.000	2	01
C	.200	.60	.10011001...	2.322	3	100
D	.150	.80	.11001100...	2.737	3	110
E	.050	.95	.11110011...	4.322	5	11110

The Shannon codeword for a letter with probability $p(x_i)$ has length $-\log_2 p(x_i)$, rounded up to the nearest integer. Since $a \leq \lceil a \rceil \leq a+1$, the average codeword length of a Shannon code therefore satisfies the sandwich bound

$$H \;\leq\; \sum_{i=1}^{k} p(x_i)\,\lceil -\log_2 p(x_i) \rceil \;\leq\; H+1.$$

We thus have an example of a coding scheme under which the average code length gets close to the Shannon bound of H bits per letter.

Exercise 11. Construct a Shannon code for each of the following lists of letter probabilities:

1. (.6, .4), 2. (.5, .5), 3. (.6, .3, .1), 4. (.4, .3, .3), 5. (.2, .2, .2, .2, .2)

Solution.

1. (0, 10), 2. (0, 1), 3. (0, 10, 1110), 4. (00, 01, 10), 5. (000, 001, 011, 100, 110)

1.8 Huffman Coding

The Shannon code frequently does not achieve the lowest possible average codeword length. In order to achieve this optimum, we use a different method which builds the code tree bottom-up by repeatedly joining the two least probable leftover tokens into a super-token that subsumes both probabilities.[6]

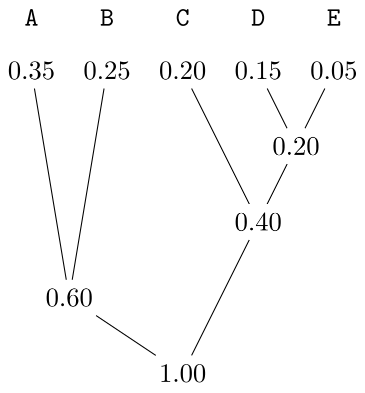
Suppose for instance we start with the following alphabet:

x	A	B	C	D	E
$p(x)$	.35	.25	.20	.15	.05

The algorithm then constructs a code tree by the following operations:

1. join D and E into a node DE with probability $.15 + .05 = .20$
2. join C and DE into a node CDE with probability $.20 + .20 = .40$
3. join A and B into a node AB with probability $.35 + .25 = .60$
4. join AB and CDE into a node ABCDE with probability $.60 + .40 = 1.00$

This results in the following code tree:



Once this tree has been constructed, it can be turned into a a prefix code by arbitrarily labeling each pair of outgoing branches with 0 and 1 (for instance by always labeling the branch with higher probability with 0). In this example, that could result in the code

A	B	C	D	E
00	01	10	110	111

A Huffman code for a variant of the English alphabet is show in Table 1.1. Note that the codeword lengths w get very close to the ideal codeword lengths $-\log_2 p(x)$ in many cases, and never exceed the ideal codeword length by more than 1 binary digit.

Exercise 12.

1. construct a Huffman code for

x	A	B	C	D
$p(x)$	.1	.2	.3	.4

[6]David A. Huffman: "A Method for the Construction of Minimum-Redundancy Codes," *Proceedings of the Institute of Radio Engineers*, 1952.

x	$p(x)$	$-\log_2 p(x)$	w	codeword
A	.0634	3.98	4	1001
B	.0135	6.21	6	011101
C	.0242	5.37	5	00011
D	.0321	4.96	5	10100
E	.0980	3.35	3	001
F	.0174	5.84	6	101111
G	.0165	5.92	6	101011
H	.0438	4.51	5	11011
I	.0552	4.18	4	0110
J	.0009	10.17	9	011100000
K	.0061	7.35	7	0111001
L	.0336	4.89	5	10110
M	.0174	5.85	6	101110
N	.0551	4.18	4	0101
O	.0622	4.01	4	1000
P	.0180	5.80	6	110100
Q	.0008	10.33	10	0111000100
R	.0470	4.41	4	0000
S	.0502	4.32	4	0100
T	.0729	3.78	4	1100
U	.0234	5.42	5	00010
V	.0075	7.06	7	0111110
W	.0156	6.00	6	011110
X	.0014	9.46	9	011100001
Y	.0160	5.97	6	101010
Z	.0005	11.04	11	01110001011
¶	.0084	6.89	7	0111111
_	.1741	2.52	3	111
’	.0019	9.06	9	011100011
,	.0117	6.42	7	1101011
.	.0109	6.52	7	1101010
?	.0003	11.56	11	01110001010

Table 1.1: A Huffman code for uppercase English with limited punctuation.

2. encode the message `DCDDAB` according to your code;
3. decode the message `0010100001` according to your code;
4. compute the expected codeword length of your code;
5. compute the entropy of the original distribution.

As we will see in the next section, Huffman codes are optimal in the sense that they achieve the smallest possible average codeword length of any uniquely decodable code. However, this does not mean that they always achieve the Shannon compression bound exactly, especially for small alphabets. For instance, a two-letter alphabet with probabilities ε and $1-\varepsilon$ can have an entropy arbitrarily close to 0, but any binary code for such an alphabet must necessarily spend 1 bit on each letter. Encoding letters from a two-letter alphabet one by one thus leads to an unavoidable waste of up to 1 bit per letter.

This waste is largest when one of the letters is very probable, and it is possible to upper-bound the average codeword length of a Huffman code as follows:

Theorem 13.[7] *The average codeword length of a Huffman code does not exceed*

[7]Robert G. Gallager: “Variations on a Theme by Huffman,” *IEEE Transactions on Information Theory*, 1978.

$H + q^* + \sigma$, where q^* is the probability of the most probable letter, and

$$\sigma \;=\; 1 - \log_2 e + \log_2 \log_2 e \;\leq\; 0.0861.$$

The mysterious number $\sigma \approx 0.086$ is the largest possible gap between $\log_2 n$ and the entropy of distribution over n possible letters under the constraint that the letter probabilities are all within a factor of 2 of each other.

1.9 The Optimality of Huffman Codes

The Huffman algorithm consists in a series of steps in which a pair of least probable nodes are merged together, and their probabilities are summed up. If you play this process in reverse, it looks like the stepwise sprouting of a binary tree, or the stepwise division of a probability mass of 1 into a series of ever finer subdivisions. For instance, if we run the example from the previous section in reverse, it would look as follows:

4. split node `ABCDE` into nodes `AB` and `CDE`
3. split node `AB` into nodes `A` and `B`
2. split node `CDE` into nodes `C` and `DE`
1. split node `DE` into nodes `D` and `E`

By construction, this backwards process always sprouts two new leaves that are less probable than any existing leaf. We will prove the optimality of the Huffman algorithm by showing that every tree generated by this process is an optimal code. This will require a series of lemmas.

Lemma 14. *An optimal code never assigns longer codewords to more probable letters.*

Proof. Otherwise we could strictly decrease the average codeword length by swapping the codewords for two letters. □

Lemma 15. *An optimal prefix code for an alphabet with nonzero and distinct letter probabilities assigns equally long codewords to the two least probable letters. (If the probabilities are nonzero but not distinct, there are two letters of minimal probability who are assigned equally long codewords.)*

Proof. Assume for a contradiction that the two least probable letters are assigned codewords of different lengths. Then the codeword assigned to the least probable letter is longer than all the other codewords. We can shorten this codeword by removing its last bit. This shortened codeword cannot be a prefix of any existing codeword because it is at least as long as any other codeword. It can also not be identical to any existing codeword since that would violate the prefix property. Since we have assumed this least probable codeword has a positive probability, removing its last bit strictly lowers the average codeword length, which contradicts the assumption of optimality. □

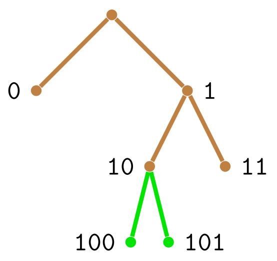
Corollary 16. *It is always possible to select an optimal code in which the two least probable codewords are siblings (i.e., differ only in their last bit).*

Proof. The two least probable codewords are equally long and of maximum length. If they are the only maximally long codewords in the code, they must also be siblings, since otherwise the code would not be optimal. If they are not already siblings, then they both have siblings of the same lengths, since otherwise the code would not be optimal; we can therefore swap one of the two codewords for the sibling of the other to produce a new code that the same average codeword length but in which the two least probable codewords are siblings. □

We will now return to the concept of code trees that sprout new leaves, and investigate how this process might change the optimality of such a code.

Definition 17. Suppose c_k is a code with codewords $c_k(x_1), \dots, c_k(x_k)$ with probabilities $p(x_1), \dots, p(x_k)$. We can **split the codeword** $c(x_k)$ by defining a new code c_{k+1} which is identical to c_k except that the codeword $c_k(x_k)$ is removed and replaced by two new codewords $c_k(x_k)\mathtt{0}$ and $c_k(x_k)\mathtt{1}$. These two new codewords can have any probabilities whose sum is $p(x_k)$.

We can of course undo a codeword splitting by merging two codewords that differ only in their last bit and adding up their probabilities.



Codeword splitting affects the average codeword length in the following way.

Theorem 18.

$$E\,|c_{k+1}(X)| \;=\; E\,|c_k(X)| + p_k(x_k)$$

Proof. The two codes yield codewords of the exact same length except in the case when the c_k returns $c_k(x_k)$ and c_{k+1} returns either $c_k(x_k)\mathtt{0}$ or $c_k(x_k)\mathtt{1}$. This event occurs with probability $p_k(x_k)$, and the codewords lengths differ by exactly one bit in that case. □

Codeword splitting does not generally preserve optimality. However, if we control the type of split, we can ensure that an optimal code remains optimal after one of its codewords is split.

Lemma 19. *If a codeword from an optimal code c_k is split, and if the two new codewords are the least probable codewords of the new code c_{k+1}, then c_{k+1} is optimal too.*

Proof. If this were not the case, there would a different code c'_{k+1} with a strictly lower average codeword length,

$$E_{k+1}\,|c_{k+1}(x_i)| \;>\; E_{k+1}\,|c'_{k+1}(x_i)|\,.$$

Since the two newly added letters are the least probable in the expanded alphabet, the code c'_{k+1} can be chosen so that their two codewords are siblings (as they already are in c_{k+1}). We then get

$$\begin{aligned} E_k\,|c_k(x_i)| + p_k(x_k) &= E_{k+1}\,|c_{k+1}(X)| \\ &> E_{k+1}\,|c'_{k+1}(X)| \;=\; E_k\,|c'_k(x_i)| + p_k(x_k), \end{aligned}$$

which contradicts the assumption that c_k is optimal. □

This last result furnishes the induction step we need in order to show that the Huffman code is optimal:

Theorem 20. *The Huffman code achieves the smallest average codeword length of any uniquely decodable code.*

Proof. Played forward, the Huffman algorithm repeatedly merges the two least probable letters on a list until everything is merged together into a single super-letter with probability 1. Played backward, it is a series of steps that repeatedly split a codeword in such a way that it is replaced by two least probable codewords.

By the previous lemma, this process always preserves optimality. Moreover, the process starts with a trivial code tree with just one codeword (the empty string) which is the optimal code for an alphabet with exactly one letter (the super-letter that subsumes the entire alphabet). It follows that every code produced by the backwards Huffman algorithm is optimal, including the final code tree for the full alphabet. □

Exercise 21. Give an example of an optimal code that becomes suboptimal when one of its codewords are split.

Exercise 22. Give a direct argument for the optimality of the Huffman algorithm for three-letter alphabets by case distinction.

1.10 Additional Exercises

Exercise 23. (Cover and Thomas, Ex. 1.1.2) Compute the entropy of a distribution with probability masses

$$1/2,\ 1/4,\ 1/8,\ 1/16,\ 1/64,\ 1/64,\ 1/64,\ 1/64.$$

Exercise 24. Rouhgly how many bits of uncertainty are created by shuffling a deck of cards?

Exercise 25. (MacKay, Ex. 15.4) How can you draw lots between three people using nothing but a fair coin? Come up with at least two different methods and compare them in terms of (a) fairness and (b) efficiency.

Exercise 26. Compute the derivative of the binary entropy function

$$H_2(q) \ = \ q \log_2 \frac{1}{q} + (1-q) \log_2 \frac{1}{1-q}$$

and plot a graph of H_2'.

Chapter 2

TUESDAY

2.1 Random Variables and Expectations

We start with an exercise:

Exercise 27. Which of the following is more probable?

1. a sum greater than 40 in 10 dice rolls
2. a sum greater than 400 in 100 dice rolls

This lecture will provide some theoretical tools that will make questions like these very easy to think about and answer. To think about this a helpful way, we'll first need to fix some terminology.

Definition 28. A **random variable** is a variable whose numeric value varies between different (randomly sampled) states of the world.

This is the conventional definition, in which we assume that a probability distribution over a set of states (e.g. playing cards) is already available, and then define the random variable as numerical values attached to those states (e.g., the value of the cards). In practice, however, we often define a distribution over numerical values directly, as when we say "Let X be drawn uniformly at random from the set $\{1, 2, \ldots, 6\}$." In that case, the states of the world already are numerical values.

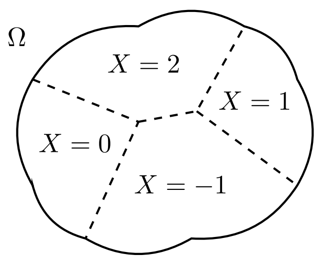


Figure 2.1.1: A random variable defined by assigning numerical values to different parts of the sample space.

A random variable with a discrete set of possible values $\{x_1, x_2, x_3, \ldots\}$ can associated with a **probability mass function** p which assigns probabilities $p(x_1), p(x_2), p(x_3), \ldots$ to those values. Note that the probability mass function p is a mapping from values to probabilities; we usually reserve a different symbol like P for the probability measure, which assigns probabilities to statements like "$X \geq 3$," which may be true in many different states of the world.

Given such a probability mass function, we can define the expected value of the random variable as follows:

Definition 29. The **expected value** of a discrete random variable X with probability mass function p is

$$EX \;=\; \sum_x x\,p(x).$$

This is the definition you would expect: a weighted average of the possible values, weighted by their probability.

2.2 Linearity of Expectations

We can define many random variables over the same sample space: for instance, we can throw two dice and let X be the sum and Y the product of number of eyes. The random variables X and Y will have a joint distribution that can be described by a **joint probability mass function** $p(x, y)$. We will abuse notation and use the same letter p as the name for the probability mass functions of X, Y, and (X, Y) since it is always clear from context which function we have in mind. We then have the **marginalization** rules

$$\begin{aligned}\sum_x p(x,y) &= p(y)\\ \sum_y p(x,y) &= p(x)\end{aligned}$$

Once we have defined a random variable over a sample space, we can also define transformations of that variable, say, $Z = X^2$. Since X has a specific numerical value in any state of the world, so does Z, and the distribution of Z can be derived from the distribution X.

The following rules help reason about such indirectly defined random variables:

Theorem 30. *For any any constant $r \in \mathbb{R}$ and any random variable X,*

$$E(rX) \;=\; r(EX).$$

For any random variables X and Y,

$$E(X+Y) \;=\; EX + EY.$$

Proof. We only prove the second rule:

$$\begin{aligned}E(X+Y) &= \sum_x\sum_y (x+y)\,p(x,y)\\ &= \sum_x\sum_y x\,p(x,y) + \sum_x\sum_y y\,p(x,y)\\ &= \sum_x x\sum_y p(x,y) + \sum_x y\sum_y p(x,y)\\ &= \sum_x x\,p(x) + \sum_x y\,p(y)\\ &= EX + EY.\end{aligned}$$

In other words, this follows from the linearity of sums, and from the marginalization rule. □

2.3 Independence

Definition 31. Suppose two random variable X and Y follow a joint probability mass function $p(x, y)$ and marginal probability mass functions $p(x)$ and $p(y)$. We say that X and Y are independent if

$$p(x, y) = p(x)\, p(y)$$

for all x and y.

Exercise 32. Under which of the following joint probability mass functions are the random variables X and Y independent?

1.

$p(x,y)$	$x=0$	$x=1$	$p(y)$
$y=0$	.4	.3	.7
$y=1$	.2	.1	.3
$p(x)$	.6	.4	1

2.

$p(x,y)$	$x=0$	$x=1$	$p(y)$
$y=0$	.42	.28	.70
$y=1$	.18	.12	.30
$p(x)$	.60	.40	1

Theorem 33. *If X and Y are independent, then*

$$E(XY) = (EX)(EY).$$

This does not necessarily hold if they are not independent.

Proof.

$$\begin{aligned} E(XY) &= \sum_x \sum_y xy\, p(x,y) = \sum_x \sum_y xy\, p(x)p(y) \\ &= \sum_x x\, p(x) \sum_y y\, p(y) = = \sum_x x\, p(x)\, EY = (EX)(EY). \end{aligned}$$

□

An example of two dependent random variables that violate the equality is a fair coin flip X, and $Y = 1 - X$; then $EX = EY = 1/2$, but $E(XY) = 0$.

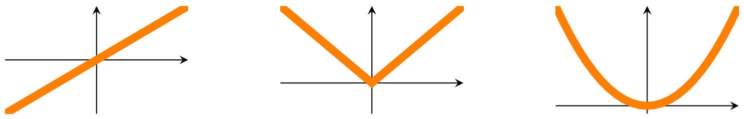
2.4 Variance

Exercise 34. Suppose X follows the distribution

x	1	2	4
$p(x)$	1/2	1/4	1/4

Compute the expected values of

1. $X - EX$, 2. $|X - EX|$, 3. $(X - EX)^2$



Which of these statistics is most suitable as a measure of dispersion?

In fact, both absolute and squared deviation are valid choices, but there are reasons of mathematical convenience that speak for the latter.

Definition 35. The **variance** of a random variable X is

$$Var(X) \; = \; E[(X - EX)^2] \; = \; E(X^2) - (EX)^2.$$

To see that these two expressions are the same, you have to remember that EX is a constant which has the same value everywhere in the sample space, so that $E(X\,EX) = (EX)(EX)$.

Exercise 36. Compute the variance of the roll of a die (i.e., a uniform distribution over the values $\{1, 2, 3, 4, 5, 6\}$).

Solution. The expectation of such a die roll X is $EX = 7/2$. The random variable $Z = (X - EX)^2$ is therefore a uniform distribution over the values $(1 - 7/2)^2, (2 - 7/2)^2, \ldots, (6 - 7/2)^2$. The average of those values is

$$\frac{25/4 + 9/4 + 1/4 + 1/4 + 9/4 + 25/4}{6} \; = \; \frac{35}{12}.$$

Exercise 37. Compute the variance of a bent coin, i.e., a binary random variable with probability masses $p(1) = q$ and $p(0) = 1 - q$.

Solution. You should get $q(1 - q)$.

2.5 Variances of Linear Combinations

Theorem 38. *For any random variable X and any constant $r \in \mathbb{R}$ and any random variable X,*

$$Var(rX) \; = \; r^2\,Var(X)$$

For independent *random variables X and Y,*

$$Var(X + Y) \; = \; Var(X) + Var(Y).$$

Note that the latter formula is not necessarily true when X and Y are not independent. As a counterexample, let $Y = 1 - X$ for some random variable X with a positive variance.

Proof. We prove only the first formula, using linearity of expectations:

$$\begin{aligned} Var(rX) &= E\left[(rX - E\{rX\})^2\right] = E\left[(rX - rEX)^2\right] \\ &= E\left[r^2(X - EX)^2\right] = r^2 E\left[(X - EX)^2\right] = r^2 Var(X). \end{aligned}$$

The second formula is most easily proven using that $Var(Z) = E(Z^2) - (EZ)^2$ and the rule $E(XY) = (EX)(EY)$ for independent X and Y. □

2.6 Means and Variances of Averages

Exercise 39. Suppose that $X_1, X_2, \dots, X_n$ are independent random variables with a common mean m and a common variance v. Let further

$$A_n = \frac{X_1 + X_2 + \cdots + X_n}{n}.$$

What is $E(A_n)$ and $Var(A_n)$?

When you inspect the solution to this exercise, you will notice that the expected value of the random average stays constant, while its variance goes to zero as n increases. In other words, the average of a very large number of random variables with a shared mean and variance almost behaves like a constant. In order to formulate this statement more precisely, we will need a few more tools.

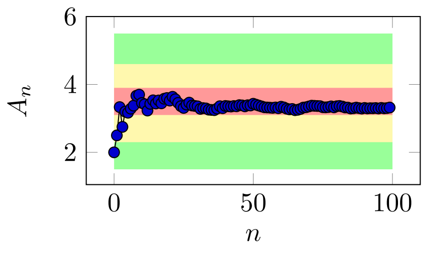


Figure 2.6.1: Convergence of averages.

Theorem 40. *Suppose that Z is non-negative random variable, in the sense that $P(Z \geq 0) = 1$. Then*

$$P(Z > z) \leq \frac{EZ}{z}.$$

This result is known as **Markov's inequality**. It says, for instance, that no more than 20% of the population can make more than five times the average income.

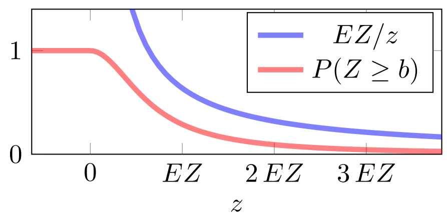


Figure 2.6.2: The survival function $P(Z \geq z)$ of a specific random variable, and its Markov bound.

Proof. Define a new random variable

$$Z^- = \begin{cases} 0 & Z \leq z \\ z & Z > z \end{cases}$$

The variable Z^- only has two possible values, so its expected value is

$$EZ^- = P(Z \leq z)\,0 + P(Z > z)\,z = P(Z > z)\,z.$$

By construction, Z^- also satisfies $Z^- \leq Z$ with probability 1. Hence $EZ^- \leq EZ$. But this implies that $P(Z > z)\,z \leq EZ$, which yields the statement of the theorem after division by z. □

Markov's inequality can be used to bound the probable deviation of a random variable from its mean. If we apply it to the nonnegative random variable $Z = (X - EX)^2$, we get the result known as known as **Chebyshev's theorem**:

$$P((X - EX)^2 > z) \;\leq\; \frac{E\left[(X - EX)^2\right]}{z} \;=\; \frac{Var(X)}{z}.$$

By combining this idea with our previous results about the mean and variance of an average, we get the **weak law of large numbers**:

Theorem 41. *Suppose $E(X_i) = m$ and $Var(X_i) = v$ for $i = 1, 2, \ldots, n$, and let $A_n = \frac{1}{n}(X_1 + X_2 + \cdots + X_n)$. Then*

$$P\left[(m - A_n)^2 > \varepsilon^2\right] \;\leq\; \frac{v}{\varepsilon^2 n}.$$

This theorem provides us with the promised quantitative statement of how the observation that averages converge to expectations. It implies that the likely deviation between an average and its mean can be bounded by a number on the order of $\varepsilon \sim 1/\sqrt{n}$.

Exercise 42. Suppose flip a coin with an unknown bias 1000 times. Compute an upper bound on the probability that the empirical frequency of heads differs from the true bias of the coin by more than $\varepsilon = 0.1$.

Solution 43. Since we don't know the bias of the coin, we don't know the variance of the coin-flipping experiment. However, for a true bias of q, the variance of the coin-flipping experiment with values 0 and 1 is $v = q(1 - q)$, which is largest for $q = 1/2$, with a value of $v = 1/4$. We can therefore upper-bound the variance of the coin flip by $v \leq 1/4$. Applying Chebyshev's inequality directly, we then get

$$P\left[(q - A_n)^2 > 0.1^2\right] \;\leq\; \frac{v}{0.1^2 1000} \;=\; \frac{v}{10} \;\leq\; \frac{1}{40}.$$

The probability that the average deviates by more than $\varepsilon = 0.1$ from the exepectation in this case can therefore by upper-bounded by $1/40$, or 2.5%.

At this point, it would be a good idea to return to the exercise at the start of this chapter and see if its solution appears obvious to you now.

2.7 Convergence of Frequencies

We now leave the world of pure probability theory and return to problems related to coding and data compression.

Exercise 44. A source produces a random word

$$X^{10} = (X_1, X_2, \ldots, X_{10}),$$

by sampling each letter independently from the distribution

x	`T`	`S`	`E`
$p(x)$	1/4	1/2	1/4

1. What is the probability of the word $X^{10} = \texttt{STETSESSES}$?
2. What is the most probable 10-letter word?
3. What is the most probable number of E's?

Each of the letters x in this example are associated with a given surprisal value, $-\log_2 p(x)$. Specifically, we have

$$\begin{aligned} -\log_2 p(\texttt{T}) &= 2 \\ -\log_2 p(\texttt{S}) &= 1 \\ -\log_2 p(\texttt{E}) &= 2 \end{aligned}$$

Because the letters are sampled independently, the probability of a word $(x_1, x_2, \ldots, x_n)$ factorizes into a product of letter probabilities,

$$p(x_1, x_2, \ldots, x_n) = p(x_1)p(x_2)\cdots p(x_n).$$

Because logarithms turn products into sums, it further follows that the surprise at seeing the word decomposes into a sum of letter surprisals:

$$\log_2 \frac{1}{p(x_1, x_2, \ldots, x_n)} = \log_2 \frac{1}{p(x_1)} + \log_2 \frac{1}{p(x_2)} + \cdots + \log_2 \frac{1}{p(x_n)}.$$

We know that the most probable word is SSS . . . S, but because there are many more words with a combination of letters, we are more likely to observe words with a lower probability mass. In fact, the weak law of large numbers implies that the relative frequencies of T, S, and E will be roughly equal to their probabilities, $\frac{1}{4} : \frac{1}{2} : \frac{1}{4}$, in a long word. Oddly enough, we are therefore unlikely to ever see the most probable word when sampling randomly.

It follows that the **sample entropy**

$$\hat{H}_n = \frac{1}{n} \log_2 \frac{1}{p(X_1, X_2, \ldots, X_n)}$$

is close to the true entropy,

$$H = E\left[\log_2 \frac{1}{p(X)}\right]$$

for large n.

Table 2.1 shows some example 10-letter words sampled from the distribution in the exercise. Note that these words frequently lie close to the true entropy of the distribution, which is 3/2.

Table 2.2 shows a similar stability for the average surprise at seeing a letter drawn from an English text. Like the synthetic example, the sample entropy of such strings of letters tends to be very predictable because the frequency of the various letters in English is very stable.

2.8 The Asymptotic Equipartition Property

We have just seen that the sample entropy

$$-\frac{1}{n} \log_2 p(X^n)$$

$X_1, X_2, \dots, X_{10}$	$\hat{H}_{10}$
TEETSTSEEE	1.8
TSSSTSTTES	1.5
TSSESETTST	1.6
SSSSESSEES	1.3
TSESSTTSST	1.5
EEESSSETSS	1.5
TESSSEESES	1.5

Table 2.1: Sample entropies from a distribution with $H = E[-\log_2 p(X)] = 1.5$ and $Var[-\log_2 p(X)] = 9.25$, and thus $E(\hat{H}_{10}) = 1.5$ and $Var(\hat{H}_{10}) = 0.925$.

$X_1, X_2, \dots, X_{40}$	$\hat{H}_{40}$
SO_SHE_BEGAN_VERY_CAUTIOUSLY:_'BUT_I_DON	4.489
U'VE_HAD!'_'OH,_I'VE_HAD_SUCH_A_CURIOUS_	4.532
F_YOU_PROVIDE_ACCESS_TO_OR_DISTRIBUTE_CO	4.325
'I_DIDN'T_KNOW_THAT_CHESHIRE_CATS_ALWAYS	4.286
LY_ONE_A-PIECE_ALL_ROUND._'BUT_SHE_MUST_	4.369
TS_EYES_WERE_GETTING_EXTREMELY_SMALL_FOR	4.321
W_LARGER,_I_CAN_REACH_THE_KEY;_AND_IF_IT	4.269

Table 2.2: A set of sample entropies for snippets of English text.

tends to be close to the entropy when n is large. Specifically, the law of large numbers allows us to state that the probability of seeing an overlarge sample entropy goes to zero as $n$increases:

$$P\left(-\frac{1}{n}\log p(X^n) \ > \ H + \varepsilon\right) \ \to \ 0.$$

By moving these terms around a bit, we can reformulate this by saying that the probability of seeing an improbable word goes to zero as n increases:

$$P\left(p(X^n) \ < \ 2^{-n(H+\varepsilon)}\right) \ \to \ 0.$$

This means that with a probability close to 1, every word we will ever encounter will have a probability $p(X^n) \geq 2^{-n(H+\varepsilon)}$. We will say that words with a probabilities above this threshold are **typical**.

Because the probability of encountering a typical word must add up to 1 or less, and because each of them has a probability of at least $2^{-n(H+\varepsilon)}$, we know that there can be at most $2^{n(H+\varepsilon)}$ typical words. This immediately gives us the following fixed-width version of Shannon's source coding theorem:

Theorem 45. *Suppose a source draws letters $X_1, X_2, X_3, \ldots$ drawn independently from a distribution with entropy H. For any $\varepsilon > 0$ and $\alpha > 0$, there is an n and an code which encodes blocks of n letters from this source using no more than $H + \varepsilon$ bits per character, such that the probability of encountering a block with no encoding is less than α.*

Proof. The proof has already been sketched above, but let us reiterate: the code in question is simple a table of equally long codewords that covers the $2^{n(H+\varepsilon)}$ most probable n-letter words. This leaves out a vast ocean of improbable sequences, but if n is chosen large enough, the law of large numbers guarantees that this ocean has a total probability below α. $\square$

The considerations above also indicate that we cannot hope to compress the source into codewords using fewer than H bits per letter: since $p(X^n)$ is close to 2^{-nH} with overwhelming probability, there is a large set of nearly equally probable codewords which subsume almost all of the probability mass. Since we need to assign a codeword to each of these 2^{nH} nearly equally probable sequences, we will need to spend nH bits per sequence.

2.9 Additional Exercises

Exercise 46. (Cover and Thomas, Ex. 3.7)

An information source produces produces pixels $X_1, X_2, X_3, \ldots$ with

$$\begin{aligned} \Pr\{X_i = \texttt{WHITE}\} &= 0.995 \\ \Pr\{X_i = \texttt{BLACK}\} &= 0.005 \end{aligned}$$

You decide to brute-force encode outputs from this source, 100 pixels at a time, by means of a table of equally long codewords. You include all sequences with three or fewer black pixels in the table and accept there will be an error in the remaining cases.

1. Compute the number of codewords you will need.
2. How many bits do you need in order to encode that many sequences? How does that compare to the theoretical minimum?
3. What are your options for improving this performance, theoretically and practically?
4. Find out how efficient your proposed encoding scheme is, and how likely it is to encounter an untabulated sequence.

Exercise 47. Devise a reasonably efficient fixed-width binary coding scheme for texts (not invidual letters) written in the alphabet $\{0, 1, 2, \ldots, 9\}$.

Exercise 48. Sketch a graph of $P(\hat{H}_n \geq b)$ as a function of b for a moderate value of n, and for a very large value of n.

Exercise 49. Construct a Shannon code for the flip a coin with bias $q = 0.8$. Compare the average codeword length for your code to the entropy of the experiment. Repeat the exercise for the experiment consisting of $n = 2$ and $n = 3$ flips of the same bent coin.

Chapter 3

WEDNESDAY

3.1 Random Processes and Entropy Rates

In previous chapters, we have proven some theorems about data compression for sequences of independent random events. However, most real-world data streams—text, sound, images—exhibit strong dependencies between consecutive events, and this can make new observations easier to predict.

To this end, we need to introduce **random processes**, which in general are probability distributions in infinite-dimensional spaces. We will focus exclusively on random processes in **discrete time**, which are probability distribution over countably infinite sequences $x_1, x_2, x_3, \ldots$. When viewed as possible values of a random process, such infinite sequences are called **sample paths**.

A discrete random process $X_1, X_2, X_3, \ldots$ can be exhaustively described by a family of conditional distributions

$$P(X_{n+1} \,|\, X_1, X_2, \ldots, X_n),$$

one for each history $X_1, X_2, \ldots, X_n$. A classic extension theorem from measure theory guarantees that such a family of finite-dimensional conditional distributions defines an essentially unique probability distribution over the infinite-dimensional space of sample paths.[1] However, all random processes we will consider will be constructed in intuitivey straightforward ways, and existence and uniqueness of the random process will not be a serious concern.

When each element in the random sequence $X_1, X_2, X_3, \ldots$ is a letter from a finite alphabet, we can define the **entropy rate** of the random process as

$$\lim_{n\to\infty} \frac{H(X_1, X_2, \ldots, X_n)}{n}$$

whenever this limit exists. Entropy rates are measured in bits per time step.

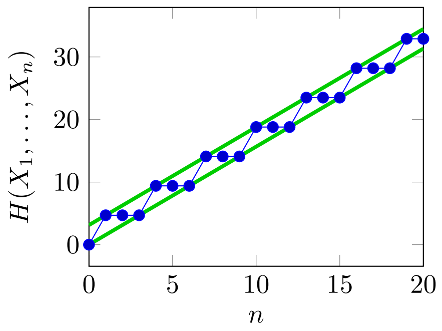
We will now look at some examples of random processes over an alphabet $\{\mathtt{A}, \mathtt{B}, \ldots, \mathtt{Z}\}$ of 26 upper-case Latin letters.

[1]This result requires the assumption of the continuity axiom of probability theory to regulate the passage to infinite dimensions. Daniell: "Integrals in An Infinite Number of Dimensions," *Annals of Mathematics* (1919), and Kolmogorov: *Grundbegriffe der Wahrscheinlichkeitsrechnung* (1933), chapters 2.2 and 3.4.

Example 50. Suppose a random process generates an infinite text by repeatedly picking a random letter and printing it three times:

`LLLEEEHHHQQQMMMQQQOOOTTTEEEYYYXXX...`

This process creates $\log_2 26 \approx 4.7$ bits of uncertainty at every third time step, and no additional uncertainty between these points:



The entropy of its initial segments are therefore sandwich-bounded by

$$\frac{n}{3}\log_2 26 \;\le\; H(X_1,X_2,\dots,X_n) \;\le\; \frac{n}{3}\log_2 26 + \frac{2}{3}\log_2 26.$$

Dividing by n, we then find that its entropy rate is

$$\lim_{n\to\infty}\frac{H(X_1,X_2,\dots,X_n)}{n} \;=\; \frac{1}{3}\log_2 26.$$

This reflects the fact that one can communicate the contents of a long initial segment $X_1, X_2, \dots, X_n$ using only a third of the bits that would be required if a new letter was sampled in every time step.

Example 51. Suppose a random process generates an infinite text by picking a single letter at random and then printing it forever:

`WWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWW...`

The initial segments of this random process satisfy

$$H(X_1,X_2,\dots,X_n) \;=\; \log_2 26,$$

since only the choice of X_1 involves any uncertainty at all. Dividing by n, we find that the entropy rate is

$$\lim_{n\to\infty}\frac{H(X_1,X_2,\dots,X_n)}{n} \;=\; 0.$$

This reflects the fact that the entire infinite sample path only contains a finite amount of uncertainty, so one can reveal the identity of an infinite number of letters by sending only a finite amont of bits.

Example 52. Suppose a random process generates an infinite text by repeatedly picking a letter at random and printing it as long as a fair coin flip comes up head:

`SSSPMMMMMDHHHKZTDUCAAAIDTTTYHHHH...`

The probability that two consecutive letters in a sample path from this process are identical is

$$\frac{1}{2} + \frac{1}{2 \cdot 26} = \frac{27}{52},$$

since this event happens either when the coin flip comes up tails (1/2), or when the coin flip comes up heads and the next letter happens to be identical to the last one ($1/2 \cdot 1/26$). Hence the next letter in the sequence is identical to the last with probability 27/52. The other 25 letters divide the remaining probability mass equally, each receiving a probability of 1/52.

The conditional distribution of the next letter in the sequence given the previous letter thus has the probability masses

$$\frac{27}{52}, \underbrace{\frac{1}{52}, \frac{1}{52}, \ldots, \frac{1}{52}}_{\text{25 options}}$$

While $H(X_1) = \log_2 26 \approx 4.7$, each letter after that thus adds another

$$\frac{27}{52} \log_2 \frac{52}{27} + \frac{25}{52} \log_2 52 \;=\; 1 + \log_2 26 - \frac{27}{52} \log_2 27 \;\approx\; 3.23$$

bits of entropy. The limiting average is therefore 3.23.

3.2 Monster Examples

We will now discuss a few monster examples that are useful to keep in mind when thinking about entropy rates.

We will start by dispensing with our usual assumption that each X_n is drawn from the same finite alphabet, instead using an infinite alphabet $\{1, 2, 3, \ldots\}$.

Example 53. Suppose each letter X_n is drawn independently at random from the set $\{1, 2, 3, \ldots, 2^n\}$. Then $H(X_n) = n$, and

$$\frac{H(X_1, X_2, \ldots, X_n)}{n} = \frac{1 + 2 + \cdots + n}{n} = \frac{n+1}{2} \to \infty.$$

No fixed budget of bits per letter will therefore be enough to transmit a message from this random process.

We will now see an example of a random process that is not **stationary**, in the sense that the random process $X_1, X_2, X_3, \ldots$ behaves very differently from its time-shifted cousins $X_n, X_{n+1}, X_{n+2}, \ldots$.

Example 54. Suppose that $X_1, X_2, X_3, \ldots$ is a random process is generated by alternating between deterministic and random runs, doubling the length of each run:

0 01 0000 01011000 0000000000000000 . . .

Then the average amount of uncertainty oscillates up and down forever,

$$\frac{H(X_1, X_2, \ldots, X_n)}{n} = \begin{cases} 2/3 & n = 3, 15, 63, 255, \ldots \\ 1/3 & n = 1, 7, 31, 127, \ldots \end{cases}$$

and this random process has no entropy rate.

Finally, let us look at an example that is not **ergodic**, meaning that it does not have the same time averages along all sample paths.

Example 55. Suppose you flip a coin and then let $X_1, X_2, X_3, \dots$ be the deterministic sequence `AAAAAA`... if the coin flip came up heads, and otherwise equal to an infinite series of random choices from an alphabet $\{\texttt{A}, \texttt{B}, \texttt{C}, \dots, \texttt{Z}\}$. Then

$$\frac{H(X_1, X_2, \dots, X_n)}{n} \;\rightarrow\; \frac{\log_2 26}{2}.$$

This reflects the fact that half of the sample paths that might occur accumulate $\log_2 26$ bits per letter, while the other half accumulate no uncertainty at all. The entropy rate is therefore well-defined, but it does not correspond to any amount of bits you will actually use: half the time, it is too much, and half the time too little.

3.3 Markov Chains

Definition 56. A random process $X_1, X_2, X_3, \dots$ is a (homogenous) **Markov chain**[2] if

$$P(X_{n+1} \,|\, X_1, X_2, \dots, X_n) \;=\; P(X_{n+1} \,|\, X_n)$$

for all n, and if the conditional distributions $P(X_{n+1} \,|\, X_n)$ are the same for all time steps n. We call the probabilities $P(X_{n+1} \,|\, X_n)$ the **transition probabilities** and the distribution $P(X_1)$ the **initial condition**.

Markov chains can be specific exhaustively described in terms of its transition probabilities and its initial condition. The transition probabilities can be specified either in the form of a **transition matrix** (with conditional distributions in the columns) or in the form of a **transition diagram**:

$$\begin{array}{c} \begin{matrix} \texttt{A} & \;\;\texttt{B} \end{matrix} \\ \begin{pmatrix} 0.7 & 0.5 \\ 0.3 & 0.5 \end{pmatrix} \end{array}$$

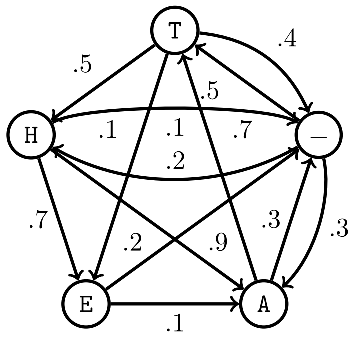
$$\begin{tikzcd} \texttt{A} \arrow[loop left, ".7"] \arrow[r, bend left, ".3"] & \texttt{B} \arrow[l, bend left, ".5"] \arrow[loop right, ".5"] \end{tikzcd}$$

The diagram is more intuitive, but the matrix form is more computationally useful: if M is a transition matrix and v_n is a vector of probability masses, then $v_{n+1} = Mv_n$ is the marginal distribution of X_{n+1} if v_n is the marginal distribution of X_n.

Markov chains are a simple but useful model of many processes, such as text, weather, or stock prices. Consider for instance the following transition probabilities:

[2]Markov: "On Trials Connected in a Chain of Unobservable Events," *Bulletin of the Imperial Saint Petersburg Academy of Sciences* (1912).

$$\begin{array}{c} \begin{array}{ccccc} \texttt{T} & \texttt{H} & \texttt{E} & \texttt{A} & \texttt{\_} \end{array} \\ \left(\begin{array}{ccccc} & & & .7 & .5 \\ .5 & & & & .2 \\ .1 & .7 & & & \\ & .2 & .1 & & 0.3 \\ .4 & .1 & .9 & .3 & \end{array}\right) \end{array}$$



A representative sample path from this random process, starting from the initial condition $X_1 = \texttt{T}$, is

```
T_ATE_T_HE_TE_THE_THE_THAT_T_TE_ATHE_AT_ATHE_T_ATHE_...
```

With a slightly larger alphabet, and transition probabilities estimated from a corpus of English text, we get sample paths like

```
MPAPTRTER WACHULLATORNTELL AR AN CO DSIS: ARON AICEN FOG S;
BUIONDE GLD RARLDIANEVOND: E; RES CTHA LATESPRR D HESALLAIND
BY MPURE ENUE AS; BIST.52092,0 PAN M GIRILERUSM: A RORSURANABER
TRUNT 17 IES BLUA WE: APREA LE ...
```

Markov chains like this one can be a good enough approximation of English text for some purposes.

3.4 Steady-State Distributions

Suppose a Markov chain $X_1, X_2, X_3, \ldots$ has initial condition v and transition matrix M. The marginal distribution of X_n can then be expressed in the form of a vector $M^n v$ whose coordinates are the letter probabilities

$$P(X_n = x_1),\ P(X_n = x_2),\ \ldots,\ P(X_n = x_k).$$

By linearity of expectations, the sum

$$w \;=\; v + Mv + M^2 v + \cdots + M^{n-1} v$$

then contains the expected number of occurrences of each letter in a text of n letters. We will now introduce some tools that can be used to compute such visiting frequencies.

Definition 57. A **stochastic vector** is a vector of nonnegative values that sum to 1. A **stochastic matrix** is a matrix whose columns are stochastic vectors.

Lemma 58. *For* $0 \leq c_1, c_2, \ldots, c_k \leq 1$

$$\sum_{i=1}^{k} c_i^2 \;\leq\; \sum_{i=1}^{k} c_i \;\leq\; \sqrt{k} \sum_{i=1}^{k} c_i^2,$$

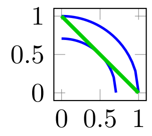
or in terms of L_2*-norms and* L_1*-norms,* $\|c\|_2^2 \leq \|c\|_1 \leq \sqrt{k}\,\|c\|_2$.

Proof. The first inequality follows from the fact that $0 \le c_i^2 \le c_i \le 1$. The second is a consequence of the Cauchy-Schwarz inequality,

$$c \cdot u \ \le\ \|c\|_2 \|u\|_2$$

with $u = (1, 1, \dots, 1)$. ☐

Viewed as vectors in k-dimensional space, the columns of a stocastic matrix all lie on the plane $\|c\|_1 = 1$. These vectors are all inside of the unit disc $\|c\|_2 \le 1$, but outside or on the border of the disc $\|c\|_2 \le 1/\sqrt{k}$.



Lemma 59. *A stochastic Eigenvector of a stochastic matrix is always associated with the Eigenvalue $\lambda = 1$.*

Proof. When M is a stochastic matrix and v is a stochastic vector, then Mv is a stochastic vector too. Hence

$$1 \ =\ \|Mv\|_1 \ =\ \lambda \|v\|_1 \ =\ \lambda.$$

☐

Lemma 60. *The Eigenvalues of a stochastic matrix are no larger than 1.*

Proof. Suppose for a contradition that $Mv = \lambda v$ with $|\lambda| > 1$. Then

$$\|M^n v\|_2 \ =\ \lambda^n \|v\|_2$$

would increase exponentially in n.

By definition, the coordinates of $M^n v$ are the dot products $v \cdot c$ of v and the columns of M^n. Since However, M^n is a stochastic matrix, $\|c\|_1 = 1$. The Cauchy-Schwarz inequality and the previous lemma therefore imply that

$$(c \cdot v)^2 \ \le\ \|c\|_2^2 \|v\|_2^2 \ \le\ \|c\|_1 \|v\|_2^2 \ =\ \|v\|_2^2.$$

Summing across coordinates, we thus find that $\|M^n v\|_2^2 \le k \|v\|_2^2$, proving that $\|M^n v\|_2$ stays below a constant bound for all n. ☐

Theorem 61 (Average Visiting Frequencies)**.** *Suppose v is a stochastic vector, and M is a stochastic matrix. Then the time-averaged visiting frequencies*

$$\frac{1}{n}(v + Mv + M^2 v + \cdots + M^{n-1} v)$$

converge to a Eigenvector of M with Eigenvalue 1.

Proof. Write v as a linear combination of the Eigenvectors $w_1, w_2, \dots, w_k$ of M:

$$w \ =\ a_1 w_1 + \cdots + a_k w_k.$$

Then

$$M^i w \ =\ a_1 \lambda_1^i w_1 + \cdots + a_k \lambda_k^i w_k,$$

and

$$\frac{1}{n} \sum_{i=0}^{n-1} M^i v \ =\ \frac{1}{n} \sum_{j=0}^{k} a_j (1 + \lambda_j + \lambda_j^2 + \cdots + \lambda_j^{n-1}) w_j.$$

There are now three different options for the nature of each Eigenvalue λ_j:

- if $\lambda_j = 1$, then
$$\frac{1}{n}(1 + \lambda + \lambda^2 + \cdots + \lambda^{n-1}) \;=\; 1$$
- if $|\lambda_j| \leq 1$, then $|1 + \lambda + \lambda^2 + \cdots + \lambda^{n-1}| \leq 2$, and
$$\frac{1}{n}(1 + \lambda + \lambda^2 + \cdots + \lambda^{n-1}) \;\to\; 0$$
- if $|\lambda_j| = 1$ but $\lambda_j \neq 1$, then the complex numbers $1, \lambda_j, \lambda_j^2, \lambda_j^3, \dots$ are evenly spaced around the complex unit circle, and
$$\frac{1}{n}(1 + \lambda + \lambda^2 + \cdots + \lambda^{n-1}) \;\to\; 0$$

Hence
$$\frac{1}{n}\sum_{i=0}^{n-1} M^i v \;\to\; \frac{1}{n}\sum_j a_j w_j$$
where the sum is over all Eigenvectors with Eigenvalue $\lambda_j = 1$. Since the left-hand side is a stochastic vector for all n, then so is its limit. Since the right-hand side is a linear combination of Eigenvectors with Eigenvalue 1, it is an Eigenvector with Eigenvalue 1. □

It follows from this theorem that we can determine the limiting visiting frequencies of a Markov chain from transition probabilities alone if there is a stochastic vector which solves the equation $Mv = v$. When there are multiple solutions, the limiting frequencies are not uniquely determined by the transition probabilities alone, and we need to take the initial condition into account.

3.5 Entropy Rates for Markov Chains

Since the next letter in a Markov chain depends only on the current letter, each letter x is associated with a particular conditional entropy
$$H(X_{n+1} \,|\, X_n = x),$$
which expresses how much uncertainty there is about the next letter when the last letter was x_n. If each letter $x_1, \dots, x_k$ in the alphabet occurs with a known limiting frequency $p^*(x)$, then the entropy rate of the process can be computed as a weighted average of the next-letter entropies:
$$\lim_{n\to\infty} \frac{H(X_1, X_2, \dots, X_n)}{n} \;=\; \sum_x p^*(x)\, H(X_{n+1} \,|\, X_n = x).$$

We will now go through some examples that illustrate this idea.

Example 62. Consider the transition probabilities

$$M = \begin{pmatrix} .5 & .5 & 0 \\ .5 & 0 & 0 \\ 0 & .5 & 1 \end{pmatrix}$$

$$\begin{tikzcd} \boxed{\text{Healty}} \arrow[loop left, ".5"] \arrow[r, bend left, ".5"] & \boxed{\text{Sick}} \arrow[l, bend left, ".5"] \arrow[r, ".5"] & \boxed{\text{Dead}} \arrow[loop right, "1"] \end{tikzcd}$$

The only stochastic vector that solves the equation $Mv = v$ is $v = (0, 0, 1)$. This reflects the fact that the process eventually gets stuck in the "dead" state with probability 1. At that point, there is no uncertainty about the next step of the sequence, and the entropy rate is

$$p^*(\text{dead})\, H(X_{n+1} \,|\, X_n = \text{dead}) \;=\; 1 \cdot 0 \;=\; 0.$$

This reflects the fact that we can reveal the identity of an infinitely long sample path by communicating only a finite amount of information, namely, what happened up to the time of the first occurrence of the "dead" state.

Example 63. Consider the transition probabilities

$$\begin{pmatrix} 0 & .5 & 0 \\ 1 & 0 & 1 \\ 0 & .5 & 0 \end{pmatrix}$$

$$\begin{tikzcd}
\mathtt{A} \arrow[r, bend left, "1"] & \mathtt{H} \arrow[l, bend left, ".5"] \arrow[r, bend right, ".5"'] & \mathtt{O} \arrow[l, bend right, "1"']
\end{tikzcd}$$

This defines a "Santa machine" that produces sample paths like

$$\mathtt{HAHOHAHAHAHOHOHOHA}\ldots$$

The only stochastic vector v that solves the equation $Mv = v$ is $v = (1/4, 1/2, 1/4)$. A sample from this Markov chain will therefore spend half its time in the "H" state, where the conditional entropy is 1 bit. The entropy rate of this process is therefore

$$p^*(\mathtt{H})\, H(X_{n+1} \,|\, X_n) \;=\; \frac{1}{2} \cdot 1 \;=\; \frac{1}{2}.$$

This reflects the fact that we need to transmit 1 bit of information for every other letter in the sample path to communicate its contents.

Example 64. Consider the transition probabilities

$$M = \begin{pmatrix} .5 & .5 & 0 \\ .5 & .5 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

$$\begin{tikzcd}
\mathtt{A} \arrow[loop left, ".5"] \arrow[r, bend left, ".5"] & \mathtt{B} \arrow[l, bend left, ".5"] \arrow[loop right, ".5"] & \mathtt{C} \arrow[loop right, "1"]
\end{tikzcd}$$

This transition matrix has a two-dimensional Eigenspace associated with the Eigenvalue 1, spanned by the stochastic vectors $(1/2, 1/2, 0)$ and $(0, 0, 1)$. The limiting visiting frequencies of this Markov chain are thus not determined uniquely from the transition matrix, but depends on the initial condition.

In particular, if a sample path begins in state `A` or state `B`, it will stay there forever and accumulate one bit of uncertainty per time step. If it begins in state `C`, it will stay there forever and not accumulate any uncertainty at at. The random processes defined by different initial conditions therefore have different entropy rates in this case.

Exercise 65. Compute the entropy rate for any Markov chain with the transition matrix

$$1. \begin{pmatrix} .6 & .9 \\ .4 & .1 \end{pmatrix} \quad 2. \begin{pmatrix} .6 & .1 \\ .4 & .9 \end{pmatrix} \quad 3. \begin{pmatrix} .7 & .2 \\ .3 & .8 \end{pmatrix} \quad 4. \begin{pmatrix} .7 & .8 \\ .3 & .2 \end{pmatrix}$$

Solution.

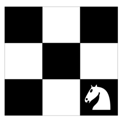
1. 0.8265 2. 0.5794 3. 0.7857 3. 0.8378

Exercise 66. In a simplified Morse alphabet, you can either send a message with a length of 2 time steps (with probability $1-\varepsilon$) or a message with a length of 3 time steps (with probability ε).

1. Compute the entropy rate of this process, measured in bits per time step,
2. Draw a graph of this entropy rate as a function of ε and approximate the value of ε that maximizes this rate.

Solution. 1. $H_2(\varepsilon)/(2+\varepsilon) \approx 4\varepsilon(1-\varepsilon)/(2+\varepsilon)$.

2. Numerically, $\varepsilon^* \approx 0.43$; algebraically, $\varepsilon^* \approx \sqrt{6}-2 \approx 0.45$.



Exercise 67. What's the entropy rate of a knight walking on a 3×3 chess board? What about a bishop? Make that the assumptions behind your computations are made explicit.

3.6 Arithmetic Coding

Exercise 68. Round the numbers $(.0, .1, .3, .4, .7, .9)$

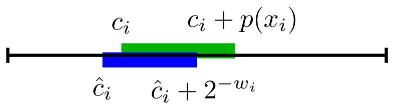
1. down to the nearest integer multiple of 1/2
2. up to the nearest integer multiple of 1/2
3. down to the neast integer multiple of 1/4
4. up to the neast integer multiple of 1/4

Solution.
1. $(0,0,0,0,\frac{1}{2},\frac{1}{2})$, 2. $(0,\frac{1}{2},\frac{1}{2},\frac{1}{2},1,1)$, 3. $(0,0,\frac{1}{4},\frac{1}{4},\frac{1}{2},\frac{3}{4})$, 4.$(0,\frac{1}{4},\frac{1}{2},\frac{1}{2},\frac{3}{4},1)$

Recall that **Shannon coding** is a method for encoding a letter x_i from a distribution with probability masses $p(x_1),\dots,p(x_k)$. It works in essence by approximating the cumulative probabilities

$$c_i \;=\; p(x_1)+p(x_2)+\cdots+p(x_{i-1})$$

from below by an integer multiple of 2^{-w_i}, where $w_i = \lceil -\log_2 p(x_i) \rceil$.



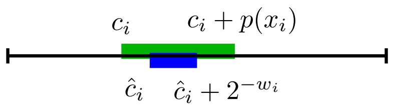
When the probability masses have already been sorted in decreasing order, each approximation $\hat{c}_1, \hat{c}_2, \ldots, \hat{c}_k$ is distinct from its neighbors on either side at its level of resolution. If, on the other hand, they have not been sorted, it is possible that $\hat{c}_i = \hat{c}_{i+1}$ in cases where $p(x_{i+1})$ is large, and c_{i+1} gets rounded down further. Fortunately, this issue can be fixed at a moderate cost:

Definition 69. The **Shannon-Fano-Elias code**[3] for a letter x_i consists of the first $w_i = \lceil -\log_2 p(x_i) \rceil + 1$ bits of the binary expansion of c_i, rounded up to the nearest integer multiple of 2^{-w_i}.

Exercise 70. Suppose an alphabet has letter probabilities $p = (.4, .1, .3, .2)$. Then the codeword lengths are $w = (3, 5, 3, 4)$, corresponding to resolutions of $\left(\frac{1}{8}, \frac{1}{32}, \frac{1}{8}, \frac{1}{16}\right)$. The cumulative probabilities are $c = (.0, .4, .5, .8)$, which at the given levels of resolution round up to

$$\hat{c} \;=\; \left(\frac{0}{8}, \frac{13}{32}, \frac{4}{8}, \frac{13}{16}\right) \;=\; (0.0, 0.40625, 0.5, 0.8125).$$

The binary expansions of these binary fractions are (`000`, `01101`, `100`, `1101`).



The approximations $\hat{c}_i$ used in the Shannon-Fano-Elias code satisfy

$$c_i \;\le\; \hat{c}_i \;<\; \hat{c}_i + 2^{-w_i} \;\le\; c_{i+1}$$

because they are upper bounds, and because the step size 2^{-w_i} is no more than half as large as $p(x_i)$. The approximations can therefore be identified from the first w_i bits in their binary expansions. Moreover, the average codeword length of the code is $H + 2$ or less.

So far, this method only provides us with yet another way of encoding a single letter. However, we will now present a method for computing the cumulative probability of all texts that come alphabetically before a given text $x_1, x_2, \ldots, x_n$. If we take $x < y$ to mean that the letter x comes alphabetically before the letter y, and let p be the probability mass function for the random text $X_1, X_2, \ldots, X_n$, then this cumulative probability can be written as

$$c(x_1, x_2, \ldots, x_n) \;=\; \sum_{x_1' < x_1} \sum_{x_2' < x_2} \cdots \sum_{x_n' < x_n} p(x_1', x_2', \ldots, x_n').$$

We then have the following recursive formula.

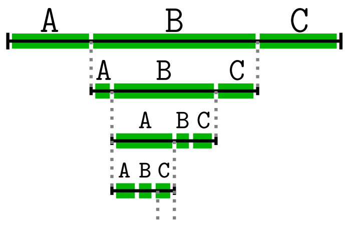
Theorem 71.

$$c(x_1, \ldots, x_{n-1}, x_n) \;=\; c(x_1, \ldots, x_{n-1}) \;+\; \sum_{x_n' < x_n} p(x_1, \ldots, x_{n-1}, x_n')$$

[3]Fano: "The Transmission of Information." Technical Report 65, Research Laboratory of Electronics, MIT (1949).

Algorithm 3.1 Arithmetic coding of a text $x_1, x_2, \dots, x_n$

```
1. Set $c = 0$ and $s = 1$
2. For for $t = 1, 2, \dots, n$:
    (a) Update $c$ and $s$ in the light of the observed letter $x_t$:
            $c \leftarrow c + s \sum_{x<x_t} p(x \mid x_1, x_2, \dots, x_{t-1})$
            $s \leftarrow s\, p(x_t \mid x_1, x_2, \dots, x_{t-1})$
    (b) While $c + s < 1/2$ or $c \geq 1/2$, print any bits that are already certain:
        i. if $c + s < 1/2$, print 0 and set $(c, s) \leftarrow (2c, 2s)$
        ii. if $c \geq 1/2$, print 1 and set $(c, s) \leftarrow (2c - 1, 2s)$
3. print the $1 + \lceil -\log_2 s \rceil$ bits of the Shannon-Fano-Elias codeword for $c$
```



Proof. Every text $x'_1, \dots, x'_n$ that comes alphabetically before $x_1, \dots, x_n$ falls into one of two classes: either

1. the initial segment $x'_1, \dots, x'_{n-1}$ comes strictly before $x_1, \dots, x_{n-1}$; or
2. the initial segment $x'_1, \dots, x'_{n-1}$ is identical to $x_1, \dots, x_{n-1}$, but $x'_n < x_n$.

The two terms of the sum compute the total probability of these two sets. □

If we combine this recursive method of computing cumulative probabilities with Shannon-Fano-Elias coding, we get a method known as **arithmetic coding.**[4] The numerically stable variant, in which a variable s is used to track the initial-segment probability $p(x_1, \dots, x_{t-1})$, is shown as Algorithm 3.1.

This variant of the algorithm also includes a step 2(b) in which any initial bits of the Shannon-Fano-Elias codeword for the text are printed as soon as they become known. If this step were left out, s would rapidly converge to zero, making the incrementing of c bump up against machine precision limits.

Exercise 72. Create a Shannon-Fano-Elias code for a random text (X_1, X_2) consisting of two independent flips of a coin with bias $q = 1/4$.

Solution 73. Assuming the probabilities are sorted in the order $\left(\frac{9}{16}, \frac{3}{16}, \frac{3}{16}, \frac{1}{16}\right)$, the code becomes $\{\texttt{00}, \texttt{1001}, \texttt{1100}, \texttt{11110}\}$.

Exercise 74. Give an example where the Shannon code that is not uniquely decodable if they probabilities are not sorted, and verify that the corresponding Shannon-Fano-Elias code is uniquely decodable.

Exercise 75 (MacKay, Ex. 8.11)**.** Consider the following two tasks:

[4]Rissanen: "Generalized Kraft Inequality and Arithmetic Coding of Strings," *IBM Journal of Research and Development* (1976).

1. predicting the next letter that comes after a snippet of text:

   `PARTICULARLY_IMPR...`

2. predicting the previous letter that came before a snippet of text:

   `NTH_FOLLOWING_THE...`

   Which of these tasks is harder—from a statistical perspective, and from a cognitive, and why?

Chapter 4

THURSDAY

4.1 Joint and Conditional Entropy

Observing one random variable may teach you something about another random variable. We will now define the **conditional entropy** $H(Y \mid X)$ as a weighted average of entropies, one for each conditional distribution $P(Y \mid X = x)$.

Definition 76. The marginal, joint, and conditional entropy of a random pair (X, Y) is defined as

$$\begin{aligned} H(X) &= E\left(\log_2 \frac{1}{p(X)}\right) \\ H(X,Y) &= E\left(\log_2 \frac{1}{p(X,Y)}\right) \\ H(Y \mid X) &= E\left(\log_2 \frac{1}{p(Y \mid X)}\right) \end{aligned}$$

where the expectations are with respect to the joint distribution of the random pair (X, Y).

Example 77. Suppose the random pair (X, Y) is distributed according to the joint probability mass function

	$X = 0$	$X = 1$	$X = 2$	
$Y = 0$	.1	.2	.1	.4
$Y = 1$	.3	.1	.2	.6
	.4	.3	.3	1

The joint entropy of the pair (X, Y) is then

$$\begin{aligned} H(X,Y) \;=\; & 0.1\log_2\frac{1}{0.1} + 0.2\log_2\frac{1}{0.2} + 0.1\log_2\frac{1}{0.1} + \\ & 0.3\log_2\frac{1}{0.3} + 0.1\log_2\frac{1}{0.1} + 0.2\log_2\frac{1}{0.2} \;\approx\; 2.45. \end{aligned}$$

The marginal entropy of X is

$$H(X) \;=\; 0.4\log_2\frac{1}{0.4} + 0.3\log_2\frac{1}{0.3} + 0.3\log_2\frac{1}{0.3} \;\approx\; 1.57,$$

and the marginal entropy of Y is

$$H(Y) \;=\; 0.4\log_2\frac{1}{0.4} + 0.6\log_2\frac{1}{0.6} \;=\; H_2(0.4) \;\approx\; 0.97.$$

Note that $H(X,Y) < H(X) + H(Y)$ in this case.

Since X can take on three different values, there are three different conditional distributions of Y given X:

	$X = 0$	$X = 1$	$X = 2$
$Y = 0$	$1/4$	$2/3$	$1/3$
$Y = 1$	$3/4$	$1/3$	$2/3$

The conditional entropy of Y given X is therefore

$$\begin{aligned} H(Y \mid X) \;=\;& 0.1\log_2\frac{4}{1} + 0.2\log_2\frac{3}{2} + 0.1\log_2\frac{2}{1} + \\ & 0.3\log_2\frac{4}{3} + 0.1\log_2\frac{3}{1} + 0.2\log_2\frac{2}{1} \;\approx\; 0.88. \end{aligned}$$

This is also the weighted average of the entropies of the three conditional distributions:

$$\begin{aligned} H(Y \mid X) &= 0.4\,H(Y \mid X = 0) + 0.3\,H(Y \mid X = 2) + 0.3\,H(Y \mid X = 2) \\ &= 0.4\,H_2(1/4) + 0.3\,H_2(2/3) + 0.3\,H_2(1/3) \;\approx\; 0.88. \end{aligned}$$

Learning the value of X thus removes about $100\% - 0.88/0.97 = 9\%$ of the uncertainty about the value of Y.

Exercise. Compute the joint, marginal, and conditional entropies of the random variables X and Y, where

1. the random pair (X, Y) is distributed according to the joint distribution

	$X = 0$	$X = 1$
$Y = 0$	.1	.4
$Y = 1$	.3	.2

2. X and Y are fair coin flips that have the same value with probability q.

Solution.

1. $H(X) \approx 0.97$, $H(X,Y) \approx 1.85$, $H(X \mid Y) \approx 0.85$, $H(Y \mid X) \approx 0.88$.
2. $H(X) = H(Y) = 1$, $H(X \mid Y) = H(Y \mid X) = H_2(q)$, $H(X,Y) = 1 + H_2(q)$.

4.2 Decomposition and Fano's Inequality

Theorem 78 (Decomposition).

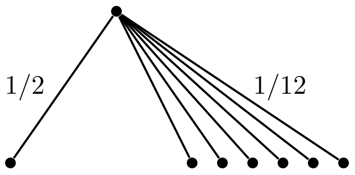
$$H(X,Y) \;=\; H(Y \mid X) + H(X).$$

Proof. Since $p(x,y) = p(y \mid x)p(x)$, we have

$$H(X,Y) \;=\; E\left(\log_2 \frac{1}{p(x,y)}\right) \;=\; E\left(\log_2 \frac{1}{p(y \mid x)} + \log_2 \frac{1}{p(x)}\right).$$

which is equal to $H(Y \mid X) + H(X)$ by the linearity of expectations. □

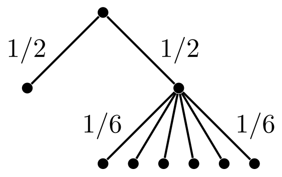
Example 79. Suppose you flip a coin to decide whether to roll a die. This experiment has seven different outcomes with probabilities $1/2, 1/12, 1/12, \ldots, 1/12$:



We can calculate the entropy of this distribution directly to be

$$\frac{1}{2}\log_2 2 + \underbrace{\frac{1}{12}\log 12 + \cdots + \frac{1}{12}\log 12}_{6 \text{ terms}} \;\approx\; 2.29.$$

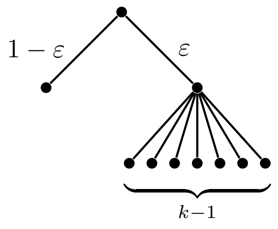
Alternatively, we can decompose this experiment into two steps, the coin flip and the possible die roll:



We can then find the total entropy of this experiment as the entropy of the coin flip plus the average of two entropies of the two conditional distributions:

$$\underbrace{H_2\left(\frac{1}{2}\right)}_{H(X)} + \underbrace{\frac{1}{2}0 + \frac{1}{2}\log_2 6}_{H(Y \mid X)} \;\approx\; 2.29.$$

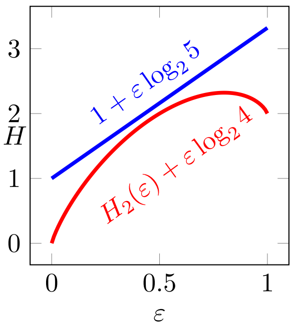
The decomposition used in this example suggests a general method for upper-bounding the entropy of a distribution by splitting off one of its point masses:



Theorem 80 (Fano's Inequality)**.** *For a random variable with k possible values, one of which has probability $p(x^*) = 1 - \varepsilon$,*

$$H(X) \leq 1 + \varepsilon \log_2 k.$$

Proof. The distribution of X can be decomposed into two random choices: whether $X = x^*$ or $X \neq x^*$; and if $X \neq x^*$, which of the remaining $k - 1$ values X takes. By decomposition, we therefore have



$$\begin{aligned} H(X) &= H_2(\varepsilon) + \varepsilon H(X \mid X \neq x^*) + \\ &\qquad (1-\varepsilon) H(X \mid X = x^*) \\ &= H_2(\varepsilon) + \varepsilon H(X \mid X \neq x^*) \\ &\leq 1 + \varepsilon \log_2 k, \end{aligned}$$

since $H_2(\varepsilon) \leq 1$ and $H(X \mid X \neq x^*) \leq \log_2(k-1) \leq \log_2 k$. □

Note that Fano's inequality is trivial when $k \leq 2$.

If x^* is the most probable value of X, then ε can also be interpreted as a minimal **error probability**: it is the probability that a forecaster will be wrong in guessing that X will be equal to its most likely value.

Because the most likely outcome cannot have a probability below $1/k$, we also have $\varepsilon \leq 1 - 1/k$ in that case. Since this implies $H_2(\varepsilon) \geq \varepsilon H_2(1/k)$, one can strengthen the usual sandwich bound

$$0 \leq H(X) \leq \log_2 k$$

on the entropy to

$$\varepsilon H_2(1/k) \leq H(X) \leq 1 + \varepsilon \log_2 k.$$

Chapter 5

FRIDAY

5.1 The Redundancy of English

Recall from the first day of the course that we observed that ordinary English text is **redundant**: it transmits more letters than are strictly necessary to get a message across. That has the consequence that letters can be lost or corrupted during transmission without loss of meaning, as long as the amount of noise isn't too great.

Consider for instance the following text:

```
AT SEVENTEEO HE HAD TRIED FPR A GOVERNMENT APPOINTMENT, BUT
HE HAD FAILED TO GET IT, BEING POOR AND FRIENRLESS, ANQ FOR
THREE YEARS HE HAD WORKED IN THE STINKING LABYRINTH OF TOE
MKNDALAY BAZAARS, CEERKING FOR THE RICE MERCHANTS APD SOME-
TIMES CTEALING. THEN WHEN HE WAS TWENTY A LUCKY STROKE OF
BLACKMAIL PUT HIM FN POSSESSION OF FOUR HUNDOED RUPEES, AND
HE WENT AT ONCE PO RANGOON ANI BOUGHT HIS WAY INTO A GOVERN-
MENT CLERKSHIP. THE JGB WAS A LUCRATIVE ONE THOUGH SHE SALARY
WAS SMALL.
```

Each letter of this text has been subjected to an equal amount of independent noise, namely, a small probability of being randomly converted into a different letter. As you can see, it is possible to recover the original message from the context even though there is no indication of which letters were corrupted.

One way of explaining this resilience is to not that there are, in a certain sense, not very many English words: the vast majority of all possible strings are not valid words. For instance, the word `THAT` is the only ordinary English word among the all sequences

```
AAAA, AAAH, AAAT, AAHA, AAHH, AAHT, AATA, AATH, AATT, AHAA,
AHAH, AHAT, AHHA, AHHH, AHHT, AHTA, AHTH, AHTT, ATAA, ATAH,
ATAT, ATHA, ATHH, ATHT, ATTA, ATTH, ATTT, HAAA, HAAH, HAAT,
HAHA, HAHH, HAHT, HATA, HATH, HATT, HHAA, HHAH, HHAT, HHHA,
HHHH, HHHT, HHTA, HHTH, HHTT, HTAA, HTAH, HTAT, HTHA, HTHH,
HTHT, HTTA, HTTH, HTTT, TAAA, TAAH, TAAT, TAHA, TAHH, TAHT,
TATA, TATH, TATT, THAA, THAH, THAT, THHA, THHH, THHT, THTA,
THTH, THTT, TTAA, TTAH, TTAT, TTHA, TTHH, TTHT, TTTA, TTTH,
TTTT
```

This also has the consequence that a given pair of English words tends to be some distance apart in terms of **edit distance** (the minimal amount of letter insertions, deletions, transpositions, or replacements needed to change the one word into the other), as shown in Table 5.1. There are some pairs that are only one edit apart, like `OUT`/`BUT` or `SHE`/`THE`, but the majority are three edits apart, making accidental confusion less likely.

	THE	AND	WAS	HIS	HAD	YOU	FOR	NOT	BUT	HER	SHE	ALL	HIM	ONE	ARE	WHO	OUT	MR.	ITS	ANY
THE	0	3	3	3	3	3	3	3	3	2	1	3	3	2	2	2	3	3	3	3
AND	3	0	3	3	2	3	3	3	3	3	3	2	3	2	2	3	3	3	3	1
WAS	3	3	0	2	2	3	3	3	3	3	3	3	3	3	3	2	3	3	2	3
HIS	3	3	2	0	2	3	3	3	3	2	3	3	1	3	3	3	3	3	2	3
HAD	3	2	2	2	0	3	3	3	3	2	3	3	2	3	3	3	3	3	3	3
YOU	3	3	3	3	3	0	2	2	3	3	3	3	3	3	3	3	2	3	3	3
FOR	3	3	3	3	3	2	0	2	3	2	3	3	3	3	3	3	3	3	3	3
NOT	3	3	3	3	3	2	2	0	2	3	3	3	3	2	3	3	2	3	3	3
BUT	3	3	3	3	3	3	3	2	0	3	3	3	3	3	3	3	1	3	3	3
HER	2	3	3	2	2	3	2	3	3	0	2	3	2	3	2	3	3	3	3	3
SHE	1	3	3	3	3	3	3	3	3	2	0	3	3	2	2	2	3	3	3	3
ALL	3	2	3	3	3	3	3	3	3	3	3	0	3	3	2	3	3	3	3	2
HIM	3	3	3	1	2	3	3	3	3	2	3	3	0	3	3	3	3	3	3	3
ONE	2	2	3	3	3	3	3	2	3	3	2	3	3	0	2	3	2	3	3	2
ARE	2	2	3	3	3	3	3	3	3	2	2	2	3	2	0	3	3	2	3	2
WHO	2	3	2	3	3	3	3	3	3	3	2	3	3	3	3	0	3	3	3	3
OUT	3	3	3	3	3	2	3	2	1	3	3	3	3	2	3	3	0	3	3	3
MR.	3	3	3	3	3	3	3	3	3	3	3	3	3	3	2	3	3	0	3	3
ITS	3	3	2	2	3	3	3	3	3	3	3	3	3	3	3	3	3	3	0	3
ANY	3	1	3	3	3	3	3	3	3	3	3	2	3	2	2	3	3	3	3	0

Table 5.1: Edit distances between some pairs of English words.

This lesson is about methods for generating codes with good properties of this kind. We will see how to select ensembles of codewords that are easy to reconstruct even when they are subjected to noise, and we will show that it is possible to transmit messages with virtually no probability of error just by adding a limited amount of redundancy.

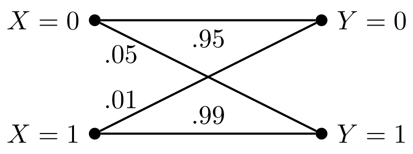
5.2 Channel and Source Models

Before we go on, let's make a few assumptions explicit, and fix some notation.

Suppose a sender transmits a signal X to a receiver, which observes a possibly corrupted signal Y. The two signals X and Y do not need to belong to the same alphabet, but in practice they often do. This noisy communication channel can be described using **channel model** which is a family $P(Y \,|\, X)$ of conditional distributions.

When both X and Y belong to finite alphabets, these conditional distributions can be described exhaustively by tabulating or drawing the conditional probability masses $p(y \,|\, x)$ for every x and y:

	$X = 0$	$X = 1$
$Y = 0$	.95	.01
$Y = 1$	.05	.99



In this example, the channel corrupts the letter $X = 0$ with 5% probability, but corrupts the letter $X = 1$ with only 1% probability.

In addition to these conditional probabilities, we will also imagine that the transmitted letter X is drawn randomly according to a distribution $P(X)$, which can be described by a list of probability masses:

$X = 0$	$X = 1$
.55	.45

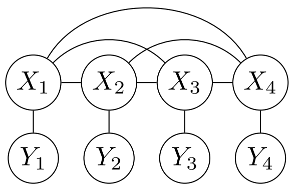
This distribution is called the **source model**. It can either be interpreted as a description of the relative frequency with which each letter in the alphabet is used, or as a model of the receiver's subjective beliefs about how likely each source letter is.

Once we have both a source model $P(X)$ and a channel model $P(Y \,|\, X)$, we have a complete probabilistic model of the random pair (X, Y). We can therefore measure marginal, joint, and conditional entropies such as for example $H(X)$ and $H(X \,|\, Y)$, the prior and posterior uncertainty about which letter was transmitted.

We will assume that the noise affects each transmitted letter independently and in the same way. If we transmit a long word $X^n = (X_1, X_2, \ldots, X_n)$, each letter in the word is therefore transmitted using the same table of conditional probabilities, leading to independent corruptions as in

$$X^{20} = \texttt{00010010111110110111}$$
$$\downarrow$$
$$Y^{20} = \texttt{00010010110110111011}$$

Note that the letters of X^n are allowed to be mutually dependent, even though we will assume that noise is is memoryless. In fact, our strategy for combatting the noise will usually be to try to create a lot of statistical dependencies between the transmitted letters, so that it's easy to detect a transmission error from context. Formally, his means that X_i and X_j may be dependent, but that Y_i and Y_j are conditionally independent given X_i and X_j for $i \neq j$. These assumptions can be illustrated in the following depedence diagram:



5.3 Channel Equipartition

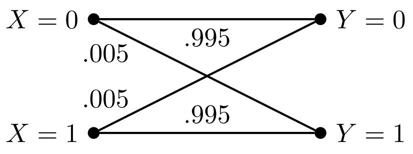
A channel model and a source distribution collectively define a joint distribution over input-output pairs (X, Y). This distribution has well-defined joint, marginal, and conditional entropies.

Example 81. Suppose a sender picks a random binary word

$$X^{1000} = (X_1, X_2, \dots, X_{1000})$$

and sends it through a channel that flips each bit with probability 0.5%:

	$X = 0$	$X = 1$
$Y = 0$	.995	.005
$Y = 1$	.005	.995



For the transmission of a single letter, we then have

$$H(Y \mid X) \;=\; H_2(.005) \;\approx\; .0454,$$

and for the transmission of the whole word,

$$H(Y^{1000} \mid X^{1000}) \;\approx\; 45.4.$$

The sender thus has about 45.4 bits of uncertainty about how the transmitted message will be received, which amounts to about $2^{45} \approx 10^{14}$ reasonably probable possibilities. This agrees with the fact that there are

$$\binom{1000}{0} + \binom{1000}{1} + \cdots + \binom{1000}{5} \;\approx\; 10^{13}$$

possibilities for how to flip 5 or fewer bits out of 1000, and about 10^{15} possibilities for how to flip 6 or fewer bits out of 1000.

This example illustrates an important application of the asymptotic equipartition property, illustrated in Figure 5.3.1:

Theorem 82. *Consider the joint distribution that arises from transmitting a random word $X^n = (X_1, X_2, \dots, X_n)$ with independent and identically distributed letters through a noisy channel. For any $\alpha > 0$ and $\varepsilon > 0$, there is then a length n and*

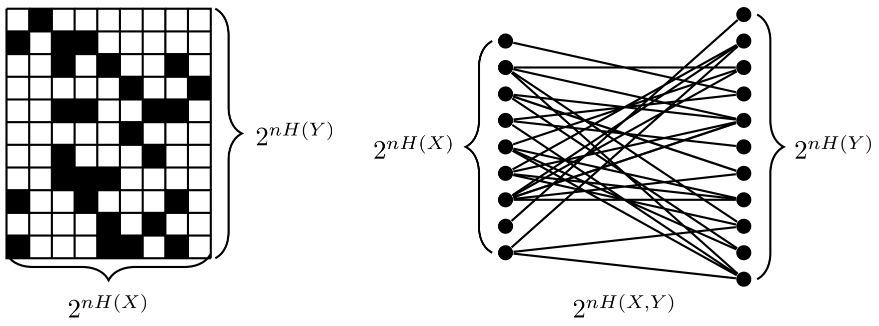


Figure 5.3.1: Typical input-output pairs for a noisy channel. On the left, the typical pairs are shown as black squares, and on the right, as connecting lines.

- *a set of no more than* $2^{n(H(X)+\varepsilon)}$ *typical strings* X^n
- *a set of no more than* $2^{n(H(Y)+\varepsilon)}$ *typical output strings* Y^n
- *a set of no more than* $2^{nH(X,Y)}$ *typical input-output pairs* (X^n, Y^n)

such that each of these sets will have probability no less than $1 - \alpha$.

Proof. This is an application of the asymptotic equipartition property to the joint and marginal distributions of the random pair (X^n, Y^n). □

5.4 Mutual Information

Definition 83. The **mutual information** between two random variables is

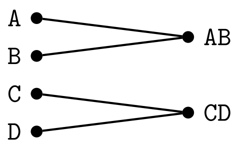
$$I(X;Y) \; = \; H(X) + H(Y) - H(X,Y).$$

Since $H(X,Y) = H(Y \mid X) + H(X)$, we can also write

$$I(X;Y) \; = \; H(Y) - H(Y \mid X).$$

The mutual information between X and Y thus measures how much uncertainty about Y is removed by observing X—and, by symmetry, how much uncertainty about X is removed by observing Y.

Example 84. Suppose X is a uniform distribution over the alphabet $\{\mathtt{A}, \mathtt{B}, \mathtt{C}, \mathtt{D}\}$, so that $H(X) = 2$. Suppose further that Y is always received as the fused letter `AB` when $X \in \{\mathtt{A}, \mathtt{B}\}$ and always received as the fused letter `CD` when $X \in \{\mathtt{C}, \mathtt{D}\}$:



Then

$$H(Y \mid X) = 0, \quad H(X,Y) = 2, \quad H(Y) = 1, \quad I(X;Y) = 1.$$

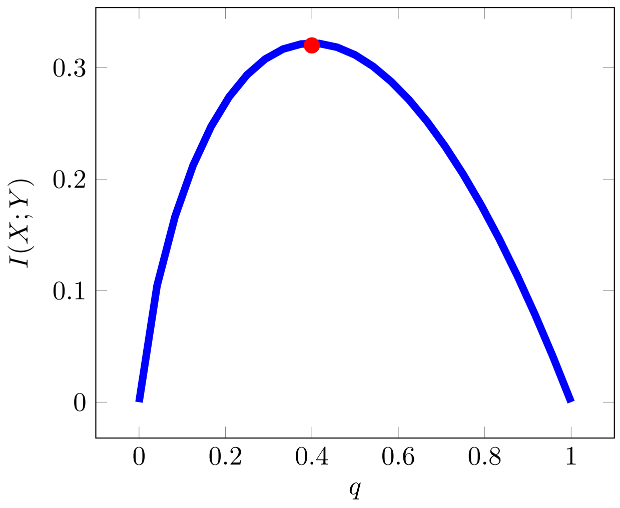


Figure 5.4.1: Mutual information for the "Z channel" for various source probabilities, showing the maximum at $q = 2/5$.

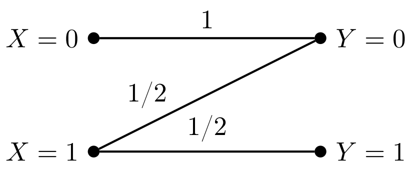
Hence learning the value of X teaches you 1 bit about Y (whether it is `AB` or `CD`). Learning the value of Y removes 1 bit of uncertainty about X (leaving 1 bit remaining).

Mutual information depends not only of the channel model $P(Y \mid X)$ but also on the source model $P(X)$. In the context of communication, it therefore makes sense to search for source distributions that maximimize information.

Definition 85. The **capacity** of a channel $P(Y \mid X)$ is the maximum of $I(X;Y)$ with respect to all possible source distributions $P(X)$.

Example 86. Consider the "Z channel" given by

	$X = 0$	$X = 1$
$Y = 0$	0	1/2
$Y = 1$	1	1/2



If the source probabilities are $1 - q$ and q, then

$$H(X) \;=\; H_2(q), \quad H(Y) = H_2(q/2), \quad H(X,Y) = H_2(q) + q.$$

Consequently, $I(X;Y) = H_2(q/2) - q$. By solving

$$\frac{\partial}{\partial q} I(X;Y) \;=\; \frac{1}{2} \log_2 \frac{1 - q/2}{q/2} - 1 \;=\; 0,$$

we find that this function is maximized at$q = 2/5$, as illustrated in Figure 5.4.1. The capacity of this channel is thus $C = H_2(1/5) - 2/5 \approx 0.32$ bits per letter.

5.5 Codes and Rates

When a message X^n is sent through a noisy channel, the received message Y^n can be typically be explained by many different hypotheses about X^n. We will now consider ways of purging certain words X^n from our language in order increase the likelihood that only a single hypothesis remains.

Definition 87. A **code of length** n is a set of k messages of n letters. The **rate** of the code is

$$\frac{1}{n}\log_2 k.$$

Rates are measured in bits per letter.

Example 88. The **threefold repetition code** is $\{\texttt{000}, \texttt{111}\}$. The rate of this code is $\frac{1}{3}\log_2 2 = 1/3$.

Example 89. The (7,4) **Hamming code**[1] is

$$\{\texttt{0000000}, \texttt{1101001}, \texttt{0101010}, \texttt{1000011}, \texttt{1001100}, \texttt{0100101}, \texttt{1100110}, \texttt{0001111},$$
$$\texttt{1110000}, \texttt{0011001}, \texttt{1011010}, \texttt{0110011}, \texttt{0111100}, \texttt{1010101}, \texttt{0010110}, \texttt{1111111}\}$$

The rate of this code is $\frac{1}{7}\log_2 16 = 4/7$.

While systematically constructed codes like these can be useful in practice, our theoretical analysis below will in fact make use of randomly selected codes.

Definition 90. A **random code** is a selection of codewords $X^n = (X_1, \ldots, X_n)$ whose letters all sampled independently from a fixed source distribution.

Note that the letters of a codeword $X^n = (X_1, \ldots, X_n)$ drawn from a fixed random code tend to be highly dependent. For instance, the code $\{\texttt{01010}, \texttt{00111}\}$ was generated by drawing two sequences of independent letters, but the letters of a word drawn from this set are not independent (e.g., the second and third letter are always different). These dependencies are random artifacts of the specific choice of code.

5.6 The Existence of Good Random Codes

We will now show that it is possible to communicate at a positive rate over a noisy channel with nearly no probability of error. In fact, we will show that most random codes of low enough rate are naturally well separated and allow nearly error-free communication.

Theorem 91. *For any $\delta > 0$, the expected error rate of a random code with any rate below $I(X;Y) - \delta$ goes to zero as $n \to \infty$.*

Proof. The theorem concerns an average across all codes, and across all choices of a message from those codes. We are going to reverse the order of the two expectations, and compute the average error rate across all messages, and across all codes contain those messages.

[1]Hamming: "Error Detecting and Error Correcting Codes," *Bell System Technical Journal*, 1950.

The marginal distribution of the transmitted message X^n is a sequence of letters drawn independently from the source distribution. The asymptotic equipartition property therefore applies, and can conclude that there are about $2^{nH(X\,|\,Y)}$ reasonable explanations for the message Y^n that the receiver will receive.

The conditional distribution for every other codeword in the code is also a sequence of letters drawn independently from the source distribution. The asymptotic equipartition property therefore applies here too, and we conclude that there are $2^{nH(X)}$ possible choices of codeword.

Consequently the probability that next codeword we sample will be a plausible but wrong explanation for the received message is

$$\frac{2^{nH(X\,|\,Y)}}{2^{nH(X)}} \;=\; 2^{-nI(X;Y)}.$$

There are $k-1$ codewords in the code in addition to the message X^n. Since the rate of the code is below $I(X;Y)-\delta$,

$$k \;\leq\; 2^{n(I(X;Y)-\delta)}.$$

The probability that one of the remaning $k-1$ codewords will be a spurious explanation is less than or equal to

$$(k-1)\,2^{-nI(X;Y)} \;\leq\; 2^{-n\delta},$$

which goes to zero exponentially in n. □

Corollary 92 (Shannon's Channel Coding Theorem)**.** *There are codes with rates arbitrarily close to the channel capacity that achieve nearly error-free communication.*

Proof. The previous theorem proves that random codes have near-zero error rates on average. Since error rates are nonnegative, this is only possible if the majority random codes have near-zero error rates. The theorem therefore indirectly proves that there are codes that achieve nearly error-free communication at a rate as high as $I(X;Y)$. By choosing a source distribution for which $I(X;Y)$ is equal to or close to the channel capacity, the corollary follows. □

Logarithms:

$$b^{\log_b x} \;=\; \log_b(b^x) \;=\; x$$

$$\log_b x \;=\; \frac{\ln x}{\ln b} \qquad \frac{\partial}{\partial x}\ln x \;=\; \frac{1}{x}$$

Binary entropy:

$$H_2(x) \quad = \quad x\log_2\frac{1}{x} + (1-x)\log_2\frac{1}{1-x} \quad \geq \quad 4x(1-x)$$

x	.1	.2	.3	.4	.5	.6	.7	.8	.9
$-\log_2 x$	3.32	2.32	1.74	1.32	1.00	0.74	0.51	0.32	0.15
$-\ln x$	2.30	1.61	1.20	0.92	0.69	0.51	0.36	0.22	0.11
$-\log_{10} x$	1.00	0.70	0.52	0.40	0.30	0.22	0.15	0.10	0.05
$-x\log_2 x$	0.33	0.46	0.52	0.53	0.50	0.44	0.36	0.26	0.14
$H_2(x)$	0.47	0.72	0.88	0.97	1.00	0.97	0.88	0.72	0.47

Jensen's inequality:

$$\begin{aligned} E\left(X^2\right) &\geq (EX)^2 \\ E\left(-\log_2 X\right) &\geq -\log_2(EX) \\ E\left(X\log_2 X\right) &\geq (EX)\log_2(EX) \end{aligned}$$

Cauchy-Schwarz:

$$v \cdot w \;\leq\; \|v\|\,\|w\|$$