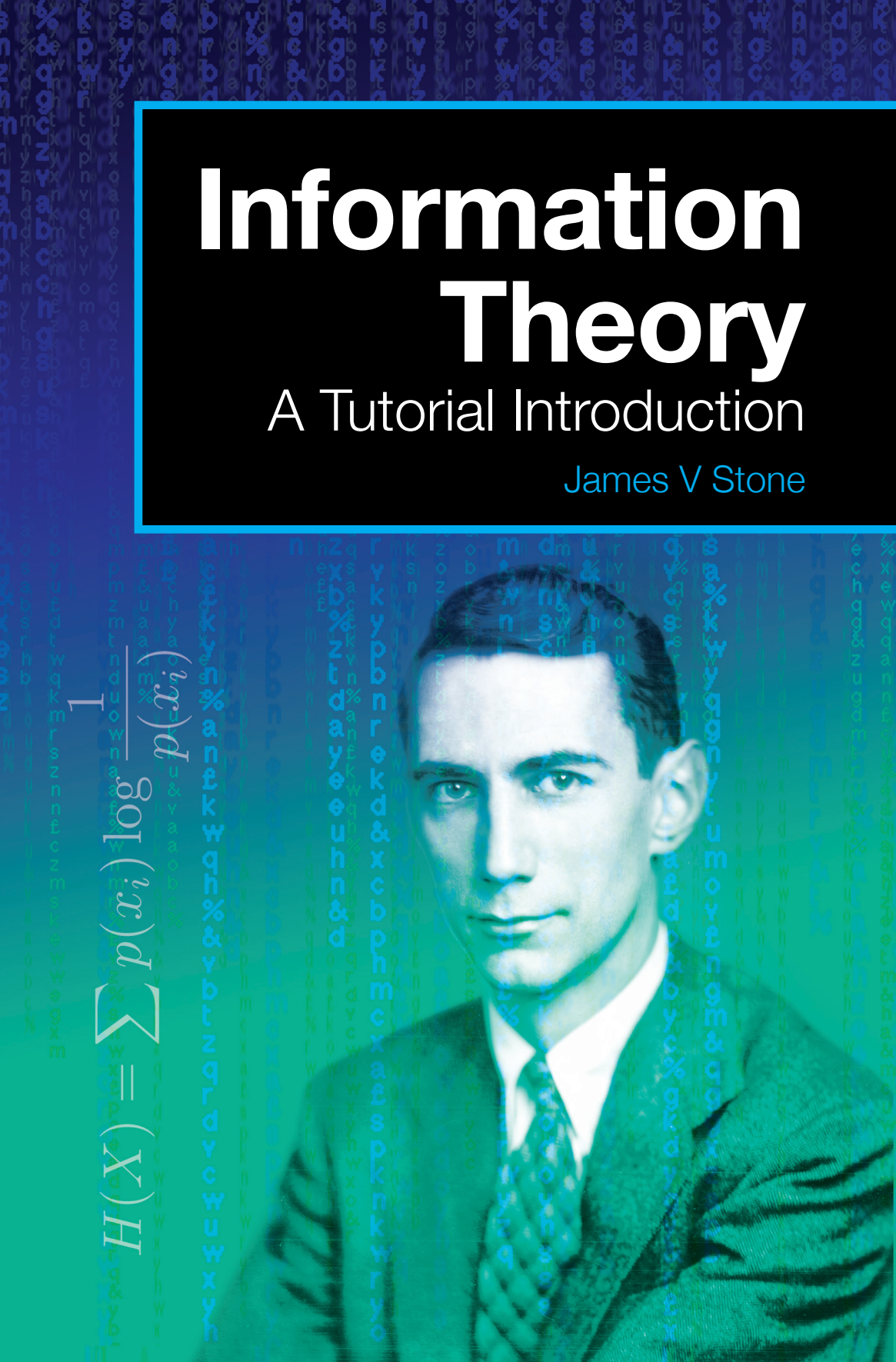
Information Theory

A Tutorial Introduction

James V Stone



Reviews of Information Theory

"Information lies at the heart of biology, societies depend on it, and our ability to process information ever more efficiently is transforming our lives. By introducing the theory that enabled our information revolution, this book describes what information is, how it can be communicated efficiently, and why it underpins our understanding of biology, brains, and physical reality. Its tutorial approach develops a deep intuitive understanding using the minimum number of elementary equations. Thus, this superb introduction not only enables scientists of all persuasions to appreciate the relevance of information theory, it also equips them to start using it. The same goes for students. I have used a handout to teach elementary information theory to biologists and neuroscientists for many years. I will throw away my handout and use this book."

Simon Laughlin, Professor of Neurobiology, Fellow of the Royal Society,
Department of Zoology, University of Cambridge, England.

"This is a really great book it describes a simple and beautiful idea in a way that is accessible for novices and experts alike. This "simple idea" is that information is a formal quantity that underlies nearly everything we do. In this book, Stone leads us through Shannons fundamental insights; starting with the basics of probability and ending with a range of applications including thermodynamics, telecommunications, computational neuroscience and evolution. There are some lovely anecdotes: I particularly liked the account of how Samuel Morse (inventor of the Morse code) pre-empted modern notions of efficient coding by counting how many copies of each letter were held in stock in a printer's workshop. The treatment of natural selection as "a means by which information about the environment is incorporated into DNA" is both compelling and entertaining. The substance of this book is a clear exposition of information theory, written in an intuitive fashion (true to Stone's observation that "rigour follows insight"). Indeed, I wish that this text had been available when I was learning about information theory. Stone has managed to distil all of the key ideas in information theory into a coherent story. Every idea and equation that underpins recent advances in technology and the life sciences can be found in this informative little book."

Professor Karl Friston, Fellow of the Royal Society. Scientific Director of the Wellcome Trust Centre for Neuroimaging,
Institute of Neurology, University College London.

Reviews of Bayes' Rule: A Tutorial Introduction

"An excellent book ... highly recommended."
CHOICE: Academic Reviews Online, February 2014.

"Short, interesting, and very easy to read, Bayes' Rule serves as an excellent primer for students and professionals ... "
Top Ten Math Books On Bayesian Analysis, July 2014.

"An excellent first step for readers with little background in the topic."
Computing Reviews, June 2014.

"The author deserves a praise for bringing out some of the main principles of Bayesian inference using just visuals and plain English. Certainly a nice intro book that can be read by any newbie to Bayes."
https://rkbookreviews.wordpress.com/, May 2015.

From the Back Cover

"Bayes' Rule explains in a very easy to follow manner the basics of Bayesian analysis."
Dr Inigo Arregui, Ramon y Cajal Researcher, Institute of Astrophysics, Spain.

"A crackingly clear tutorial for beginners. Exactly the sort of book required for those taking their first steps in Bayesian analysis."
Dr Paul A. Warren, School of Psychological Sciences, University of Manchester.

"This book is short and eminently readable. It introduces the Bayesian approach to addressing statistical issues without using any advanced mathematics, which should make it accessible to students from a wide range of backgrounds, including biological and social sciences."
Dr Devinder Sivia, Lecturer in Mathematics, St John's College, Oxford University, and author of Data Analysis: A Bayesian Tutorial.

"For those with a limited mathematical background, Stone's book provides an ideal introduction to the main concepts of Bayesian analysis."
Dr Peter M Lee, Department of Mathematics, University of York. Author of Bayesian Statistics: An Introduction.

"Bayesian analysis involves concepts which can be hard for the uninitiated to grasp. Stone's patient pedagogy and gentle examples convey these concepts with uncommon lucidity."
Dr Charles Fox, Department of Computer Science, University of Sheffield.

Information Theory

A Tutorial Introduction

James V Stone

Title: Information Theory: A Tutorial Introduction
Author: James V Stone

First Edition, 2015.
Typeset in LaTeX 2_ε.
Cover design: Stefan Brazzo.
Second printing.

ISBN 978–0–9563728–5–7

The front cover depicts Claude Shannon (1916-2001).

For Nikki

Suppose that we were asked to arrange the following in two categories – *distance, mass, electric force, entropy, beauty, melody.* I think there are the strongest grounds for placing entropy alongside beauty and melody ...

Eddington A, The Nature of the Physical World, 1928.

Contents

Preface

This book is intended to provide a coherent and succinct account of information theory. In order to develop an intuitive understanding of key ideas, new topics are first presented in an informal tutorial style before being described more formally. In particular, the equations which underpin the mathematical foundations of information theory are introduced on a need-to-know basis, and the meaning of these equations is made clear by explanatory text and diagrams.

In mathematics, rigour follows insight, and not *vice versa*. Kepler, Newton, Fourier and Einstein developed their theories from deep intuitive insights about the structure of the physical world, which requires, but is fundamentally different from, the raw logic of pure mathematics. Accordingly, this book provides insights into *how* information theory works, and *why* it works in that way. This is entirely consistent with Shannon's own approach. In a famously brief book, Shannon prefaced his account of information theory for continuous variables with these words:

> We will not attempt in the continuous case to obtain our results with the greatest generality, or with the extreme rigor of pure mathematics, since this would involve a great deal of abstract measure theory and would obscure the main thread of the analysis. ... The occasional liberties taken with limiting processes in the present analysis can be justified in all cases of practical interest.
> Shannon C and Weaver W, 1949[50].

In a similar vein, Jaynes protested that:

> Nowadays, if you introduce a variable x without repeating the incantation that it is some set or 'space' X, you are accused of dealing with an undefined problem ...
> Jaynes ET and Bretthorst GL, 2003[26].

Even though this is no excuse for sloppy mathematics, it is a clear recommendation that we should not mistake a particular species of pedantry for mathematical rigour. The spirit of this liberating and somewhat cavalier approach is purposely adopted in this book, which is intended to provide insights, rather than incantations, regarding how information theory is relevant to problems of practical interest.

MatLab and Python Computer Code

It often aids understanding to be able to examine well-documented computer code which provides an example of a particular calculation or method. To support this, MatLab and Python code implementing key information-theoretic methods can be found online. The code also reproduces some of the figures in this book.
MatLab code can be downloaded from here:
http://jim-stone.staff.shef.ac.uk/BookInfoTheory/InfoTheoryMatlab.html
Python code can be downloaded from here:
http://jim-stone.staff.shef.ac.uk/BookInfoTheory/InfoTheoryPython.html

PowerPoint Slides of Figures

Most of the figures used in this book are available for teaching purposes as a pdf file and as PowerPoint slides. These can be downloaded from
http://jim-stone.staff.shef.ac.uk/BookInfoTheory/InfoTheoryFigures.html

Corrections

Please email corrections to j.v.stone@sheffield.ac.uk.
A list of corrections can be found at
http://jim-stone.staff.shef.ac.uk/BookInfoTheory/Corrections.html

Acknowledgments

Thanks to John de Pledge, John Porrill, Royston Sellman, and Steve Snow for interesting discussions on the interpretation of information theory, and to John de Pledge for writing the Python code.

For reading draft versions of this book, I am very grateful to Óscar Barquero-Pérez, Taylor Bond, David Buckley, Jeremy Dickman, Stephen Eglen, Charles Fox, Nikki Hunkin, Danielle Matthews, Guy Mikawa, Xiang Mou, John de Pledge, John Porrill, Royston Sellman, Steve Snow, Tom Stafford, Paul Warren and Stuart Wilson. Shashank Vatedka deserves a special mention for checking the mathematics in a final draft of this book. Thanks to Caroline Orr for meticulous copy-editing and proofreading.

Online code for estimating the entropy of English was adapted from code by Neal Patwari (MatLab) and Clément Pit-Claudel (Python).

For permission to use the photograph of Claude Shannon, thanks to the Massachusetts Institute of Technology.

Jim Stone.

Chapter 1

What Is Information?

> Most of the fundamental ideas of science are essentially simple, and may, as a rule, be expressed in a language comprehensible to everyone.
> Einstein A and Infeld L, 1938.

1.1. Introduction

The universe is conventionally described in terms of physical quantities such as mass and velocity, but a quantity at least as important as these is *information*. Whether we consider computers[30], evolution[2;19], physics[15], artificial intelligence[9], quantum computation[46], or the brain[17;43], we are driven inexorably to the conclusion that their behaviours are largely determined by the way they process information.

Figure 1.1. Claude Shannon (1916-2001).

In 1948, Claude Shannon published a paper called *A Mathematical Theory of Communication*[48]. This paper heralded a transformation in our understanding of information. Before Shannon's paper, information had been viewed as a kind of poorly defined miasmic fluid. But after Shannon's paper, it became apparent that information is a well-defined and, above all, *measurable* quantity.

Shannon's paper describes a subtle theory which tells us something fundamental about the way the universe works. However, unlike other great theories such as the Darwin–Wallace theory of evolution, information theory is not simple, and it is full of caveats. But we can disregard many of these caveats provided we keep a firm eye on the physical interpretation of information theory's defining equations. This will be our guiding principle in exploring the theory of information.

1.2. Information, Eyes and Evolution

Shannon's theory of information provides a mathematical definition of information, and describes precisely how much information can be communicated between different elements of a system. This may not sound like much, but Shannon's theory underpins our understanding of how signals and noise are related, and why there are definite limits to the rate at which information can be communicated within *any* system, whether man-made or biological. It represents one of the few examples of a single theory creating an entirely new field of research. In this regard, Shannon's theory ranks alongside those of Darwin–Wallace, Newton, and Einstein.

When a question is typed into a computer search engine, the results provide useful information but it is buried in a sea of mostly useless data. In this internet age, it is easy for us to appreciate the difference between information and data, and we have learned to treat the information as a useful 'signal' and the rest as distracting 'noise'. This experience is now so commonplace that technical phrases like 'signal to noise ratio' are becoming part of everyday language. Even though most people are unaware of the precise meaning of this phrase, they have an intuitive grasp of the idea that 'data' means a combination of (useful) signals and (useless) noise.

The ability to separate signal from noise, to extract information from data, is crucial for modern telecommunications. For example, it allows a television picture to be compressed to its bare information bones and transmitted to a satellite, then to a TV, before being decompressed to reveal the original picture on the TV screen.

This type of scenario is also ubiquitous in the natural world. The ability of eyes and ears to extract useful signals from noisy sensory data, and to package those signals efficiently, is the key to survival[51]. Indeed, the *efficient coding hypothesis*[5;8;43;55] suggests that the evolution of sense organs, and of the brains that process data from those organs, is primarily driven by the need to minimise the energy expended for each bit of information acquired from the environment. More generally, a particular branch of brain science, *computational neuroscience*, relies on information theory to provide a benchmark against which the performance of neurons can be objectively measured.

On a grander biological scale, the ability to separate signal from noise is fundamental to the Darwin–Wallace theory of evolution by natural selection[12]. Evolution works by selecting the individuals best suited to a particular environment so that, over many generations, information about the environment gradually accumulates within the gene pool. Thus, natural selection is essentially a means by which information about the environment is incorporated into DNA (deoxyribonucleic acid). And it seems likely that the rate at which information is incorporated into DNA is accelerated by an age-old biological mystery, sex. These and other applications of information theory are described in Chapter 9.

1.3. Finding a Route, Bit by Bit

Information is usually measured in *bits*, and one bit of information allows you to choose between two equally probable alternatives. The word bit is derived from *bi*nary dig*it* (i.e. a zero or a one). However, as we shall see, bits and binary digits are fundamentally different types of entities.

Imagine you are standing at the fork in the road at point A in Figure 1.2, and that you want to get to the point marked D. Note that this figure represents a bird's-eye view, which you do not have; all you have

is a fork in front of you, and a decision to make. If you have no prior information about which road to choose then the fork at A represents two equally probable alternatives. If I tell you to go left then you have received one bit of information. If we represent my instruction with a *binary digit* (0=left and 1=right) then this binary digit provides you with one bit of information, which tells you which road to choose.

Now imagine that you stroll on down the road and you come to another fork, at point B in Figure 1.2. Again, because you have no idea which road to choose, a binary digit (1=right) provides one bit of information, allowing you to choose the correct road, which leads to the point marked C.

Note that C is one of four possible interim destinations that you could have reached after making two decisions. The two binary digits that allow you to make the correct decisions provided two bits of information, allowing you to choose from four (equally probable) possible alternatives; 4 happens to equal $2 \times 2 = 2^2$.

A third binary digit (1=right) provides you with one more bit of information, which allows you to again choose the correct road, leading to the point marked D.

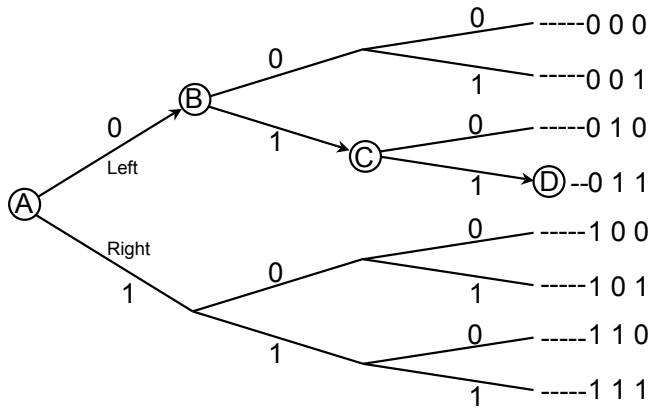


Figure 1.2. How many roads must a man walk down? For a traveller who does not know the way, each fork in the road requires one bit of information to make a correct decision. The 0s and 1s on the right-hand side summarise the instructions needed to arrive at each destination; a left turn is indicated by a 0 and a right turn by a 1.

There are now eight roads you could have chosen from when you started at A, so three binary digits (which provide you with three bits of information) allow you to choose from eight equally probable alternatives; 8 happens to equal $2 \times 2 \times 2 = 2^3 = 8$.

The decision taken at A excluded half of the eight possible destinations shown in Figure 1.2 that you could have arrived at. Similarly, the decision taken at each successive fork in the road halved the number of remaining possible destinations.

A Journey of Eight Alternatives

Let's summarise your journey in terms of the number of equally probable alternatives:

> If you have 1 bit of information then you can choose between 2 equally probable alternatives (i.e. $2^1 = 2$).
>
> If you have 2 bits of information then you can choose between 4 equally probable alternatives (i.e. $2^2 = 4$).
>
> Finally, if you have 3 bits of information then you can choose between 8 equally probable alternatives (i.e. $2^3 = 8$).

We can restate this in more general terms if we use n to represent the number of forks, and m to represent the number of final destinations. If you have come to n forks, then you have effectively chosen from

$$m \quad = \quad 2^n \text{ final destinations.} \tag{1.1}$$

Because the decision at each fork requires one bit of information, n forks require n bits of information, which allow you to choose from 2^n equally probable alternatives.

There is a saying that "a journey of a thousand miles begins with a single step". In fact, a journey of a thousand miles begins with a single decision: the direction in which to take the first step.

Key point. One bit is the amount of information required to choose between two *equally probable* alternatives.

Binary Numbers

We could label each of the eight possible destinations with a decimal number between 0 and 7, or with the equivalent *binary number*, as in Figure 1.2. These decimal numbers and their equivalent binary representations are shown in Table 1.1. Counting in binary is analogous to counting in decimal. Just as each decimal digit in a decimal number specifies how many 1s, 10s, 100s (etc) there are, each binary digit in a binary number specifies how many 1s, 2s, 4s (etc) there are. For example, the value of the decimal number 101 equals the number of 100s (i.e. 10^2), plus the number of 10s (i.e. 10^1), plus the number of 1s (i.e. 10^0):

$$(1 \times 100) + (0 \times 10) + (1 \times 1) \quad = \quad 101. \tag{1.2}$$

Similarly, the value of the binary number 101 equals the number of 4s (i.e. 2^2), plus the number of 2s (i.e. 2^1), plus the number of 1s (i.e. 2^0):

$$(1 \times 4) + (0 \times 2) + (1 \times 1) \quad = \quad 5. \tag{1.3}$$

The binary representation of numbers has many advantages. For instance, the binary number that labels each destination (e.g. 011) explicitly represents the set of left/right instructions required to reach that destination. This representation can be applied to any problem that consists of making a number of two-way (i.e. binary) decisions.

Logarithms

The complexity of any journey can be represented either as the number of possible final destinations or as the number of forks in the road which must be traversed in order to reach a given destination. We know that as the number of forks increases, so the number of possible destinations also increases. As we have already seen, if there are three forks then there are $8 = 2^3$ possible destinations.

Decimal	0	1	2	3	4	5	6	7
Binary	000	001	010	011	100	101	110	111

Table 1.1. Decimal numbers and their equivalent binary representations.

Viewed from another perspective, if there are $m = 8$ possible destinations then how many forks n does this imply? In other words, given eight destinations, what power of 2 is required in order to get 8? In this case, we know the answer is $n = 3$, which is called the *logarithm* of 8. Thus, $3 = \log_2 8$ is the number of forks implied by eight destinations.

More generally, the logarithm of m is the power to which 2 must be raised in order to obtain m; that is, $m = 2^n$. Equivalently, given a number m which we wish to express as a logarithm,

$$n = \log_2 m. \tag{1.4}$$

The subscript $_2$ indicates that we are using logs to the base 2 (all logarithms in this book use base 2 unless stated otherwise). See Appendix C for a tutorial on logarithms.

A Journey of $\log_2(8)$ Decisions

Now that we know about logarithms, we can summarise your journey from a different perspective, in terms of bits:

> If you have to choose between 2 equally probable alternatives (i.e. 2^1) then you need $1 (= \log_2 2^1 = \log_2 2)$ bit of information.

> If you have to choose between 4 equally probable alternatives (i.e. 2^2) then you need $2 (= \log_2 2^2 = \log_2 4)$ bits of information.

> If you have to choose between 8 equally probable alternatives (i.e. 2^3) then you need $3 (= \log_2 2^3 = \log_2 8)$ bits of information.

More generally, if you have to choose between m equally probable alternatives, then you need $n = \log_2 m$ bits of information.

Key point. If you have n bits of information, then you can choose from $m = 2^n$ equally probable alternatives. Equivalently, if you have to choose between m equally probable alternatives, then you need $n = \log_2 m$ bits of information.

1.4. A Million Answers to Twenty Questions

Navigating a series of forks in the road is, in some respects, similar to the game of '20 questions'. In this game, your opponent chooses a word (usually a noun), and you (the astute questioner) are allowed to ask 20 questions in order to discover the identity of this word. Crucially, each question must have a yes/no (i.e. binary) answer, and therefore provides you with a maximum of one bit of information.

By analogy with the navigation example, where each decision at a road fork halved the number of remaining destinations, each question should *halve* the number of remaining possible words. In doing so, each answer provides you with exactly one bit of information. A question to which you already know the answer is a poor choice of question. For example, if your question is, "Is the word in the dictionary?", then the answer is almost certainly, "Yes!", an answer which is predictable, and which therefore provides you with no information.

Conversely, a well-chosen question is one to which you have no idea whether the answer will be yes or no; in this case, the answer provides exactly one bit of information. The cut-down version of '20 questions' in Figure 1.3 shows this more clearly.

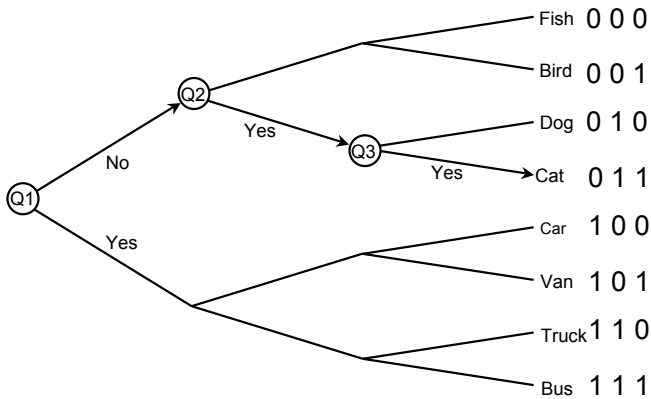


Figure 1.3. The game of '20 questions', here abbreviated to '3 questions'. Given an opponent who has one of eight words in mind, each yes/no question halves the number of remaining possible words. Each binary number on the right summarises the sequence of answers required to arrive at one word (no=0 and yes=1).

In this game, your opponent has a vocabulary of exactly eight words, and you know which words they are. Your first question (Q1) could be, "Is it inanimate?", and the answer should halve the number of possible words to four, leading you to your second question (Q2). If your second question (Q2) is, "Is it a mammal?", then the answer should again halve the number of possible words, leading to your third question (Q3). By the time you arrive at Q3, there are just two possible words left, and after you have asked the third question (e.g. "Is it 'cat'?"), your opponent's yes/no response leads you to the correct answer. In summary, you have asked three questions, and excluded all but one out of eight possible words.

More realistically, let's assume your opponent has the same vocabulary as you do (most of us have similar vocabularies, so this assumption is not entirely unreasonable). Specifically, let's assume this vocabulary contains exactly 1,048,576 words. Armed with this knowledge, each question can, in principle, be chosen to halve the number of remaining possible words. So, in an ideal world, your first question should halve the number of possible words to 524,288. Your next question should halve this to 262,144 words, and so on. By the time you get to the 19th question there should be just two words left, and after the 20th question, there should be only one word remaining.

The reason this works out so neatly is because 20 questions allow you to choose from exactly $1,048,576 = 2^{20}$ equally probable words (i.e. about one million). Thus, the 20 bits of information you have acquired with your questioning provide you with the ability to narrow down the range of possible words from about 1 million to just one. In other words, 20 questions allow you to find the correct word out of about a million possible words.

Adding one more question would not only create a new game, '21 questions', it would also double the number of possible words (to about 2 million) that you could narrow down to one. By extension, each additional question allows you to acquire up to one more bit of information, and can therefore double the initial number of words. In principle, a game of '40 questions' allows you to acquire 40 bits of information, allowing you to find one out of $2^{40} \approx 10^{12}$ words.

In terms of the navigation example, 40 bits would allow you to navigate 40 forks in the road, and would therefore permit you to choose one out of about a trillion possible routes. So the next time you arrive at your destination after a journey that involved 40 decisions, remember that you have avoided arriving at a trillion-minus-one incorrect destinations.

1.5. Information, Bits and Binary Digits

Despite the fact that the word *bit* is derived from *binary digit*, there is a subtle, but vital, difference between them. A binary digit is the value of a binary variable, where this value can be either a 0 or a 1, but a binary digit is not information *per se*. In contrast, a bit is a definite *amount of information*. Bits and binary digits are different types of entity, and to confuse one with the other is known as a *category error*.

To illustrate this point, consider the following two extreme examples. At one extreme, if you already know that you should take the left-hand road from point A in Figure 1.2 and I show you the binary digit 0 (=left), then you have been given a binary digit but you have gained no information. At the other extreme, if you have no idea about which road to choose and I show you a 0, then you have been given a binary digit and you have also gained one bit of information. Between these extremes, if someone tells you there is a 71% probability that the left-hand road represents the correct decision and I subsequently confirm this by showing you a 0, then this 0 provides you with less than one bit of information (because you already had some information about which road to choose). In fact, when you receive my 0, you gain precisely half a bit of information (see Section 5.8). Thus, even though I cannot give you a half a binary digit, I can use a binary digit to give you half a bit of information.

The distinction between binary digits and bits is often ignored, with Pierce's book[40] being a notable exception. Even some of the best textbooks use the terms 'bit' and 'binary digit' interchangeably. This does not cause problems for experienced readers as they can interpret the term 'bit' as meaning a binary digit or a bit's worth of information according to context. But for novices the failure to respect this distinction is a source of genuine confusion.

Sadly, in modern usage, the terms bit and binary digit have become synonymous, and MacKay (2003)[34] proposed that the unit of information should be called the *Shannon*.

Key point. A bit is the *amount of information* required to choose between two equally probable alternatives (e.g. left/right), whereas a binary digit is the *value of a binary variable*, which can adopt one of two possible values (i.e. 0/1).

1.6. Example 1: Telegraphy

Suppose you have just discovered that if you hold a compass next to a wire, then the compass needle changes position when you pass a current through the wire. If the wire is long enough to connect two towns like London and Manchester, then a current initiated in London can deflect a compass needle held near to the wire in Manchester.

You would like to use this new technology to send messages in the form of individual letters. Sadly, the year is 1820, so you will have to wait over 100 years for Shannon's paper to be published. Undeterred, you forge ahead. Let's say you want to send only upper-case letters, to keep matters simple. So you set up 26 electric lines, one per letter from A to Z, with the first line being A, the second line being B, and so on. Each line is set up next to a compass which is kept some distance from all the other lines, to prevent each line from deflecting more than one compass.

In London, each line is labelled with a letter, and the corresponding line is labelled with the same letter in Manchester. For example, if you want to send the letter D, you press a switch on the fourth line in London, which sends an electric current to Manchester along the wire which is next to the compass labelled with the letter D. Of course, lines fail from time to time, and it is about 200 miles from London to Manchester, so finding the location of the break in a line is difficult and expensive. Naturally, if there were fewer lines then there would be fewer failures.

With this in mind, Cooke and Wheatstone devised a complicated two-needle system, which could send only 23 different letters. Despite

the complexity of their system, it famously led to the arrest of a murderer. On the first of January 1845, John Tawell poisoned his mistress, Sarah Hart, in a place called Salt Hill in the county of Berkshire, before escaping on a train to Paddington station in London. In order to ensure Tawell's arrest when he reached his destination, the following telegraph was sent to London:

> A MURDER HAS GUST BEEN COMMITTED AT SALT HILL AND THE SUSPECTED MURDERER WAS SEEN TO TAKE A FIRST CLASS TICKET TO LONDON BY THE TRAIN WHICH LEFT SLOUGH AT 742 PM HE IS IN THE GARB OF A KWAKER ...

The unusual spellings of the words JUST and QUAKER were a result of the telegrapher doing his best in the absence of the letters J, Q and Z in the array of 23 letters before him. As a result of this telegram, Tawell was arrested and subsequently hanged for murder. The role of Cooke and Wheatstone's telegraph in Tawell's arrest was widely reported in the press, and established the practicality of telegraphy.

In the 1830s, Samuel Morse and Alfred Vail developed the first version of (what came to be known as) the *Morse code*. Because this specified each letter as dots and dashes, it could be used to send messages over a single line.

An important property of Morse code is that it uses short *codewords* for the most common letters, and longer codewords for less common letters, as shown in Table 1.2. Morse adopted a simple strategy to find out which letters were most common. Reasoning that newspaper

A ●-	J ●- - -	S ●●●
B - ●●●	K - ● -	T -
C - ●- ●	L ●- ●●	U ●●-
D - ●●	M - -	V ●●-
E ●	N - ●	W ●- -
F ●●- ●	O - - -	X - ●●-
G - - ●	P ●- - ●	Y - ●- -
H ●●●●	Q - - ●-	Z - - ●●
I ●●	R ●- ●	

Table 1.2. Morse code. Common letters (e.g. E) have the shortest codewords, whereas rare letters (e.g. Z) have the longest codewords.

printers would have only as many copies of each letter as were required, he went to a printer's workshop and counted the copies of each letter. As a result, the most common letter E is specified as a single dot, whereas the rare J is specified as a dot followed by three dashes.

The ingenious strategy adopted by Morse is important because it enables efficient use of the communication channel (a single wire). We will return to this theme many times, and it raises a fundamental question: how can we tell if a communication channel is being used as efficiently as possible?

1.7. Example 2: Binary Images

The internal structure of most images is highly predictable. For example, most of the individual *pic*ture *el*ements or *pixels* in the image of stars in Figure 1.4 are black, with an occasional white pixel, a star. Because almost all pixels are black, it follows that most pairs of adjacent pixels are also black, which makes the image's internal structure predictable. If this picture were taken by the orbiting Hubble telescope then its predictable structure would allow it to be efficiently transmitted to Earth.

Suppose you were in charge of writing the computer code which conveys the information in Figure 1.4 from the Hubble telescope to Earth. You could naively send the value of each pixel; let's call this method A. Because there are only two values in this particular image (black and white), you could choose to indicate the colour black with the binary digit 0, and the colour white with a 1. You would therefore need to send as many 0s and 1s as there are pixels in the image. For example, if the image was 100×100 pixels then you would need to send ten thousand 0s or 1s for the image to be reconstructed on Earth. Because almost all the pixels are black, you would send sequences of hundreds of 0s interrupted by the occasional 1. It is not hard to see that this is a wasteful use of the expensive satellite communication channel. How could it be made more efficient?

Another method consists of sending only the locations of the white pixels (method B). This would yield a code like $[(19, 13), (22, 30), \dots]$, where each pair of numbers represents the row and column of a white pixel.

Figure 1.4. The night sky. Each pixel contains one of just two values.

Yet another method consists of concatenating all of the rows of the image, and then sending the number of black pixels that occur before the next white pixel (method C). If the number of black pixels that precede the first white pixel is 13 and there are 9 pixels before the next white pixel, then the first row of the image begins with $0000000000000100000000001\ldots$, and the code for communicating this would be $[13, 9, \ldots]$, which is clearly more compact than the 24 binary digits which begin the first row of the image.

Notice that method A consists of sending the image itself, whereas methods B and C do not send the image, but they do send all of the *information* required to reconstruct the image on Earth. Crucially, the end results of all three methods are identical, and it is only the efficiency of the methods that differs.

In fact, whether A, B, or C is the most efficient method depends on the structure of the image. This can be seen if we take an extreme example consisting of just one white pixel in the centre of the image. For this image, method A is fairly useless, because it would require 10,000 binary values to be sent. Method B would consist of two numbers, $(50, 50)$, and method C would consist of a single number, $5,050$. If we ignore the brackets and commas then we end up with four decimal digits for both methods B and C. So these methods seem to be equivalent, at least for the example considered here.

For other images, with other structures, different *encoding methods* will be more or less efficient. For example, Figure 1.5 contains just two grey-levels, but these occur in large regions of pure black or pure

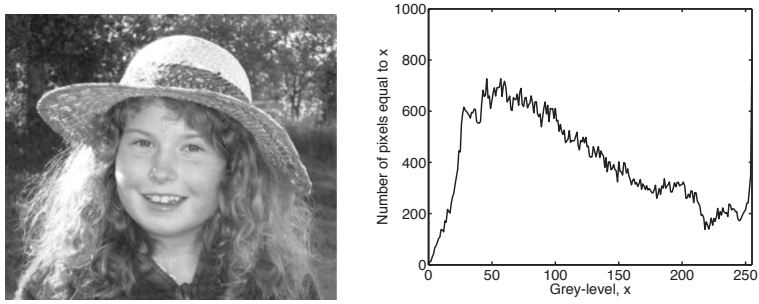
Figure 1.5. In a binary image, each pixel has 1 out of 2 possible grey-levels.

white. In this case, it seems silly to use method B to send the location of every white pixel, because so many of them occur in long runs of white pixels. This observation makes method C seem to be an obvious choice – but with a slight change. Because there are roughly equal numbers of black and white pixels which occur in regions of pure black or pure white, we could just send the number of pixels which precede the next change from black to white or from white to black. This is known as *run-length encoding*.

To illustrate this, if the distance from the first black pixel in the middle row to the first white pixel (the girl's hair) is 87 pixels, and the distance from there to the next black pixel is 31 pixels, and the distance to the next white pixel is 18 pixels, then this part of the image would be encoded as $[87, 31, 18, \dots]$. Provided we know the method used to encode an image, it is a relatively simple matter to reconstruct the original image from the encoded image.

1.8. Example 3: Grey-Level Images

Suppose we wanted to transmit an image of 100×100 pixels, in which each pixel has more than two possible grey-level values. A reasonable number of grey-levels turns out to be 256, as shown in Figure 1.6a. As before, there are large regions that look as if they contain only one grey-level. In fact, each such region contains grey-levels which are similar, but not identical, as shown in Figure 1.7. The similarity between nearby pixel values means that adjacent pixel values are not *independent* of



(a) (b)

Figure 1.6. Grey-level image. (a) An image in which each pixel has one out of 256 possible grey-levels, between 0 and 255, each of which can be represented by a binary number with 8 binary digits (e.g. 255=11111111). (b) Histogram of grey-levels in the picture.

each other, and that the image has a degree of *redundancy*. How can this observation be used to encode the image?

One method consists of encoding the image in terms of the differences between the grey-levels of adjacent pixels. For brevity, we will call this *difference coding*. (More complex methods exist, but most are similar in spirit to this simple method.) In principle, pixel differences could be measured in any direction within the image, but, for simplicity, we concatenate consecutive rows to form a single row of 10,000 pixels, and then take the difference between adjacent grey-levels. We can see the result of difference coding by 'un-concatenating' the rows to reconstitute an image, as shown in Figure 1.8a, which looks like a badly printed version of Figure 1.6a. As we shall see, both images contain the same amount of information.

If adjacent pixel grey-levels in a given row are similar, then the difference between the grey-levels is close to zero. In fact, a histogram of difference values shown in Figure 1.8b shows that the most common difference values are indeed close to zero, and only rarely greater than ± 63. Thus, using difference coding, we could represent almost every one of the 9,999 difference values in Figure 1.8a as a number between -63 and $+63$.

In those rare cases where the grey-level difference is larger than ± 63, we could list these separately as each pixel's location (row and column

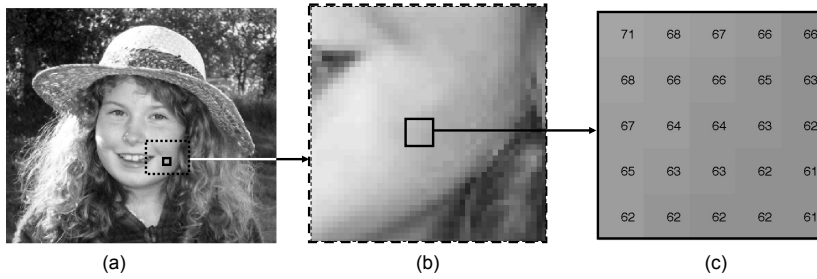


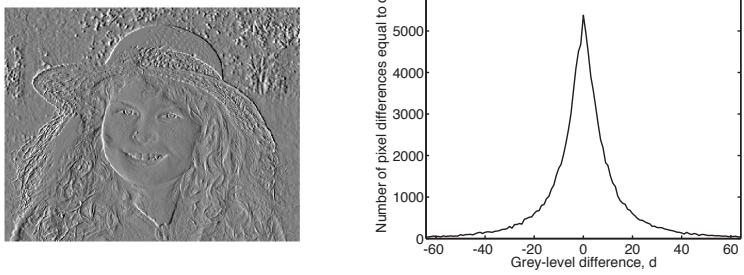
Figure 1.7. Adjacent pixels tend to have similar grey-levels, so the image has a large amount of redundancy, which can be used for efficient encoding. (a) Grey-level image. (b) Magnified square from a. (c) Magnified square from b, with individual pixel grey-levels indicated.

as 2×7 binary digits), and its grey-level (8 binary digits). Most coding procedures have special 'housekeeping' fragments of computer code to deal with things like this, but these account for a negligible percentage of the total storage space required. For simplicity, we will assume that this percentage is zero.

At first, it is not obvious how difference coding represents any saving over simply sending the value of each pixel's grey-level. However, because these differences are between -63 and $+63$, they span a range of 127 different values, i.e. $[-63, -62, \ldots, 0, \ldots, 62, 63]$. Any number in this range can be represented using seven binary digits, because $7 = \log 128$ (leaving one spare value).

In contrast, if we were to send each pixel's grey-level in Figure 1.6a individually, then we would need to send 10,000 grey-levels. Because each grey-level could be any value between 0 and 255, we would have to send eight binary digits ($8 = \log 256$) for each pixel.

Once we have encoded an image into 9,999 pixel grey-level differences $(d_1, d_2, \ldots, d_{9999})$, how do we use them to reconstruct the original image? If the difference d_1 between the first pixel grey-level x_1 and the second pixel grey-level x_2 is, say, $d_1 = (x_2 - x_1) = 10$ grey-levels and the grey-level of x_1 is 5, then we obtain the original grey-level of x_2 by adding 10 to x_1; that is, $x_2 = x_1 + d_1$ so $x_2 = 5 + 10 = 15$. We then continue this process for the third pixel ($x_3 = x_2 + d_2$), and so on. Thus, provided we know the grey-level of the first pixel in the original image (which can be encoded as eight binary digits), we can use the



(a) (b)

Figure 1.8. Difference coding. (a) Each pixel grey-level is the difference between adjacent horizontal grey-level values in Figure 1.6a (grey = zero difference). (b) Histogram of grey-level differences between adjacent pixel grey-levels in Figure 1.6a. Only differences between ±63 are plotted.

pixel grey-level differences to recover the grey-level of every pixel in the original image. The fact that we can reconstruct the original image (Figure 1.6a) from the grey-level differences (Figure 1.8a) proves that they both contain exactly the same amount of *information*.

Let's work out the total saving from using this difference coding method. The naive method of sending all pixel grey-levels, which vary between 0 and 255, would need eight binary digits per pixel, requiring a total of 80,000 binary digits. Using difference coding we would need seven binary digits per difference value, making a total of 70,000 binary digits. Therefore, using difference coding provides a saving of 10,000 binary digits, or 12.5%.

In practice, a form of difference coding is used to reduce the amount of data required to transmit voices over the telephone, where it is known as *differential pulse code modulation*. Using the differences between consecutive values, a voice signal which would otherwise require eight binary digits per value can be transmitted with just five binary digits.

As we shall see in subsequent chapters, a histogram of data values (e.g. image grey-levels) can be used to find an upper bound for the average amount of information each data value could convey. Accordingly, the histogram (Figure 1.6b) of the grey-levels in Figure 1.6a defines an upper bound of 7.84 bits/pixel. In contrast, the histogram (Figure 1.8b) of the grey-level differences in Figure 1.8a defines an upper bound of just 5.92 bits/pixel.

Given that the images in Figures 1.6a and 1.8a contain the same amount of information, and that Figure 1.8a contains no more than 5.92 bits/pixel, it follows that Figure 1.6a cannot contain more than 5.92 bits/pixel either. This matters because Shannon's work guarantees that if each pixel's grey-level contains an average of 5.92 bits of information, then we should be able to represent Figure 1.6a using no more than 5.92 binary digits per pixel. But this still represents an upper bound. In fact, the smallest number of binary digits required to represent each pixel is equal to the amount of information (measured in bits) implicit in each pixel. So what we really want to know is: how much information does each pixel contain?

This is a hard question, but we can get an idea of the answer by comparing the amount of computer memory required to represent the image in two different contexts (for simplicity, we assume that each pixel has eight binary digits). First, in order to display the image on a computer screen, the value of each pixel occupies eight binary digits, so the bigger the picture, the more memory it requires to be displayed. Second, a compressed version of the image can be stored on the computer's hard drive using an average of less than eight binary digits per pixel (e.g. by using the difference coding method above). Consequently, storing the (compressed) version of an image on the hard drive requires less memory than displaying that image on the screen. In practice, image files are usually stored in compressed form with the method used to compress the image indicated by the file name extension (e.g. '.jpeg').

The image in Figure 1.6a is actually 344 by 299 pixels, where each pixel grey-level is between 0 and 255, which can be represented as eight binary digits (because $2^8 = 256$), or one *byte*. This amounts to a total of 102,856 pixels, each of which is represented on a computer screen as one byte. However, when the file containing this image is inspected, it is found to contain only 45,180 bytes; the image in Figure 1.6a can be compressed by a factor of $2.28 (= 102856/45180)$ without any loss of information. This means that the information implicit in each pixel, which requires eight binary digits for it to be displayed on a screen,

can be represented with about four binary digits on a computer's hard drive.

Thus, even though each pixel can adopt any one of 256 possible grey-levels, and is displayed using eight binary digits of computer memory, the grey-level of each pixel can be stored in about four binary digits. This is important, because it implies that each set of eight binary digits used to display each pixel in Figure 1.6a contains an average of only four bits of information, and therefore each binary digit contains only *half a bit* of information. At first sight, this seems like an odd result. But we already know from Section 1.5 that a binary digit can represent half a bit, and we shall see later (especially in Chapter 5) that a fraction of a bit is a well-defined quantity which has a reasonably intuitive interpretation.

1.9. Summary

From navigating a series of forks in the road, and playing the game of '20 questions', we have seen how making binary choices requires information in the form of simple yes/no answers. These choices can also be used to choose from a set of letters, and can therefore be used to send typed messages along telegraph wires.

We found that increasing the number of choices from two (forks in the road) to 26 (letters) to 256 (pixel grey-levels) allowed us to transmit whole images down a single wire as a sequence of binary digits. In each case, the redundancy of the data in a message allowed it to be compressed before being transmitted. This redundancy emphasises a key point: a binary digit does not necessarily provide one bit of information. More importantly, a binary digit is *not* the same type of entity as a bit of information.

So, what is information? It is what remains after every iota of natural redundancy has been squeezed out of a message, and after every aimless syllable of noise has been removed. It is the unfettered essence that passes from computer to computer, from satellite to Earth, from eye to brain, and (over many generations of natural selection) from the natural world to the collective gene pool of every species.

Chapter 2

Entropy of Discrete Variables

> Information is the resolution of uncertainty.
> Shannon C, 1948.

2.1. Introduction

Now that we have an idea of the key concepts of information theory, we can begin to explore its inner workings on a more formal basis. But first, we need to establish a few ground rules regarding *probability*, *discrete variables* and *random variables*. Only then can we make sense of *entropy*, which lies at the core of information theory.

2.2. Ground Rules and Terminology

Probability

We will assume a fairly informal notion of probability based on the number of times particular events occur. For example, if a bag contains 40 white balls and 60 black balls then we will assume that the probability of reaching into the bag and choosing a black ball is the same as the proportion, or *relative frequency*, of black balls in the bag (i.e. $60/100 = 0.6$). From this, it follows that the probability of an event (e.g. choosing a black ball) can adopt any value between zero and one, with zero meaning it definitely will not occur, and one meaning it definitely will occur. Finally, given a set of mutually exclusive events (such as choosing a ball, which has to be either black or white), the probabilities of those events must add up to one (e.g. $0.4 + 0.6 = 1$). See Appendix F for an overview of the rules of probability.

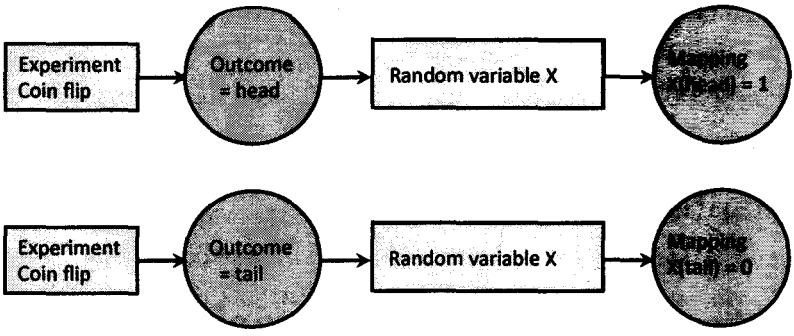


Figure 2.1. Random variables. A random variable translates the *outcome* of an experiment to an *outcome value*. Top: flipping a coin yields a head, which is mapped by the random variable X to the outcome value $X(head) = 1$, usually written as $X = 1$. Bottom: flipping a coin yields a tail, which is mapped to the outcome value $X(tail) = 0$, usually written as $X = 0$.

Discrete Variables

Elements of a set that are clearly separated from each other, like a list of integers, are called *discrete*, and the variables used to represent them are called *discrete variables*. The distribution of probability values of a discrete variable is called a *probability function*. It can be represented as a bar graph, as in Figure 2.2.

In contrast, elements which are packed together so densely that there is no space between them, like the points on a line, are represented by *continuous variables* (see Chapter 5), which have distributions called *probability density functions* (see Appendix D). We will usually refer to both probability functions and probability density functions as *probability distributions* in this text.

Random Variables

A *random variable* is used to refer to a special type of quantity; it can be either discrete or continuous. The value of a discrete random variable can be considered as a measurement made on a physical outcome of an experiment in which the number of different possible outcomes is discrete, for example as shown in Figure 2.1. In contrast, the value of a continuous random variable can be considered as a measurement made on a physical outcome of an experiment in which the values of

the possible outcomes are continuous, such as temperature. The crucial point is that these outcomes are subject to a degree of randomness.

The idea of a random variable was devised for historical reasons. Although they share the name 'variable', random variables are not the same as the variables used in algebra, like the x in $3x + 2 = 5$, where the variable x has a definite, but unknown, value that we can solve for.

A random variable is represented by an upper-case letter, such as X. An experiment consisting of a coin flip has two possible physical outcomes, a head x_h and a tail x_t, which define the *alphabet*

$$A_x = \{x_h, x_t\}. \tag{2.1}$$

The *sample space* of the random variable X is the set of all possible experiment outcomes. For example, if an experiment consists of three coin flips then each time the experiment is run we obtain a sequence of three outcomes (e.g. (x_h, x_t, x_t)), which is one out of the eight possible sequences of three outcomes that comprise the sample space.

The value of the random variable is a mapping from the experiment outcome to a numerical *outcome value*. Thus, strictly speaking, a random variable is not a variable at all, but is really a *function* which maps outcomes to outcome values. In our experiment, this function maps the coin flip outcome to the number of heads observed:

$$X(x_h) = 1, \tag{2.2}$$

$$X(x_t) = 0. \tag{2.3}$$

Thus, a random variable (function) takes an *argument* (e.g. x_h or x_t), and returns an outcome value (e.g. 0 or 1). An equivalent, and more conventional, notation for defining a random variable is

$$X = \begin{cases} 0, & \text{if the outcome is a tail,} \\ 1, & \text{if the outcome is a head.} \end{cases}$$

For brevity, the different possible values of a random variable can also be written in terms of the outcomes,

$$X = x_h, \tag{2.4}$$

$$X = x_t, \tag{2.5}$$

or in terms of the outcome values,

$$X = 1, \tag{2.6}$$

$$X = 0. \tag{2.7}$$

In many experiments, different physical outcomes have different probabilities, so that different outcome values of X also have different probabilities. If the same experiment is repeated infinitely many times then the frequency with which different values of X occur defines the *probability distribution* $p(X)$ of X. There are only two possible outcomes for a coin, so $p(X)$ consists of just two probabilities,

$$p(X) = \{p(X = x_h), p(X = x_t)\}, \tag{2.8}$$

which is usually written as

$$p(X) = \{p(x_h), p(x_t)\}. \tag{2.9}$$

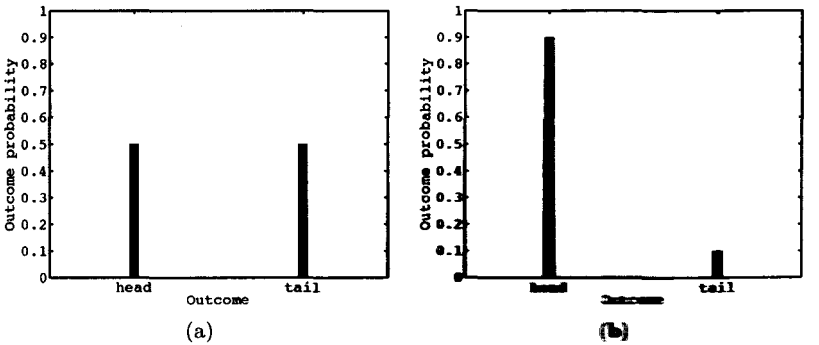


Figure 2.2. The probability distributions of two coins. **(a)** Probability distribution $p(X) = \{p(x_h), p(x_t)\}$ of a fair coin which lands heads up with a probability of $p(x_h) = 0.5$. **(b)** Probability distribution of an unfair coin which lands heads up with a probability of $p(x_h) = 0.9$.

For a fair coin both probabilities are equal to 0.5, i.e.

$$p(x_h) = 0.5, \qquad p(x_t) = 0.5, \tag{2.10}$$

so the probability distribution is

$$\begin{aligned} p(X) &= \{p(x_h), p(x_t)\} && (2.11)\\ &= \{0.5, 0.5\}, && (2.12) \end{aligned}$$

as shown in Figure 2.2a. In contrast, if a coin is so badly bent that it lands heads up 90% of the time then

$$p(x_h) = 0.9, \qquad p(x_t) = 0.1, \tag{2.13}$$

and the probability distribution of this coin is (see Figure 2.2b)

$$\begin{aligned} p(X) &= \{p(x_h), p(x_t)\} && (2.14)\\ &= \{0.9, 0.1\}. && (2.15) \end{aligned}$$

It is worth noting that the probabilities associated with different values of a discrete random variable like X vary continuously between zero and one, even though the values of X are discrete.

The subtle distinction between an *outcome* x and an *outcome value* $X(x)$ is sometimes vital, but in practice we only need to distinguish between them if the numbers of outcomes and outcome values are not equal (e.g. the two-dice example in Section 3.5). For example, suppose we roll a die, and we define the random variable X to be 0 when the outcome is an odd number and 1 when the outcome is an even number, so that

$$X = \begin{cases} 0, & \text{if the outcome } x \text{ is 1,3, or 5,} \\ 1, & \text{if the outcome } x \text{ is 2,4, or 6.} \end{cases}$$

In this case, the number of outcomes is six, but the number of outcome values is just two.

In the majority of cases, where we do not need to distinguish between outcomes and outcome values, we will use the lower case symbol x to

represent them both. The terms outcome and outcome value are unique to this book.

All of this may seem like a lot of work just to set the value of a random variable to 0 or 1, but these definitions are key to making sense of more complicated scenarios. A brief tutorial on probability distributions is given in Appendix D.

Key point. A random variable X is a function that maps each outcome x of an experiment (e.g. a coin flip) to a number $X(x)$, which is the outcome value of x. If the outcome value of x is 1 then this may be written as $X = 1$, or as $x = 1$.

Information Theory Terminology

For historical reasons, information theory has its own special set of terms. We have encountered some of these terms before, but here, and in Figure 2.3, we give a more detailed account.

First, we have a *source* which generates *messages*. A message is an ordered sequence of k *symbols*

$$\mathbf{s} = (s_1, \ldots, s_k), \tag{2.16}$$

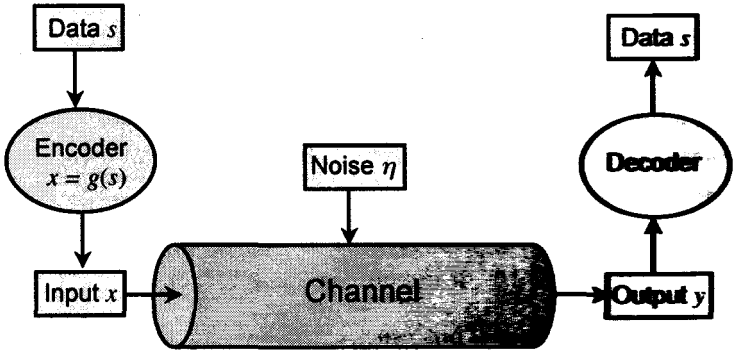


Figure 2.3. The communication channel. A message $\mathbf{s}$ is encoded as codewords $\mathbf{x}$ before being transmitted through a channel, which may corrupt the encoded message by adding noise η to produce outputs $\mathbf{y} = \mathbf{x} + \eta$. A receiver decodes the output $\mathbf{y}$ to recover inputs $\mathbf{x}$, which are then interpreted as a message $\mathbf{s}$.

where each symbol can be a number or a letter corresponding to the value of a random variable. Notice that the entire sequence **s** is represented by a letter in bold typeface, and its symbols are enclosed in round brackets.

Each symbol is an outcome value of a random variable S which can adopt any value from an alphabet of α (alpha) different symbols,

$$A_s = \{s_1, \ldots, s_\alpha\}. \tag{2.17}$$

The probability of each symbol being generated by the source is defined by the probability distribution

$$p(S) = \{p(s_1), \ldots, p(s_\alpha)\}, \tag{2.18}$$

where, by definition, the sum of $p(s)$ values must add up to one,

$$\sum_{i=1}^{\alpha} p(s_i) = 1. \tag{2.19}$$

The summation symbol $\sum$ is explained in Appendix B.

A *communication channel* is used to transmit data from its input to its output. If these data are transmitted from input to output without error then they have been successfully *communicated.* Before being transmitted, each message **s** is transformed by an *encoder*, which we can represent as a generic function g, into the *channel input* $\mathbf{x} = g(\mathbf{s})$, which is a sequence of *codewords*

$$\mathbf{x} = (x_1, \ldots, x_n), \tag{2.20}$$

where each codeword is the value of a random variable X which can adopt any one of m different values from the *codebook*

$$A_x = \{x_1, \ldots, x_m\}. \tag{2.21}$$

The probability of each codeword is defined by the probability distribution

$$p(X) = \{p(x_1), \ldots, p(x_m)\}. \tag{2.22}$$

We may choose to transmit a message as it is, so that the message and the transmitted codewords are identical (i.e. $\mathbf{x} = \mathbf{s}$). However, several symbols may be combined into a single codeword (see Section 3.4), or the message may be compressed by removing its natural redundancy. Consequently, the number of codewords in an encoded message $\mathbf{x} = g(\mathbf{s})$ may not match the number of symbols in the message $\mathbf{s}$.

If the form of compression allows the message to be decompressed perfectly then the compression is *lossless* (see Section 3.5), but if some information is discarded during compression then the message cannot be recovered exactly, and the compression is *lossy*.

A *code* is a list of symbols and their corresponding codewords. It can be envisaged as a simple look-up table (e.g. Table 2.1).

In order to ensure that the encoded message can withstand the effects of a noisy communication channel, some redundancy may be added to codewords before they are transmitted (see Section 4.7).

Transmitting a message $\mathbf{s}$ encoded as the codewords $\mathbf{x}$ produces the channel outputs

$$\mathbf{y} = (y_1, \ldots, y_n). \tag{2.23}$$

Each output is the value of a random variable Y, which can adopt any one of m different values

$$A_y = \{y_1, \ldots, y_m\}. \tag{2.24}$$

If the channel is noisy then the output y_j may be different from the codeword x_j that was transmitted.

Symbol	Codeword	Symbol	Codeword
$s_1 = 3$	$x_1 = 000$	$s_5 = 15$	$x_5 = 100$
$s_2 = 6$	$x_2 = 001$	$s_6 = 18$	$x_6 = 101$
$s_3 = 9$	$x_3 = 010$	$s_7 = 21$	$x_7 = 110$
$s_4 = 12$	$x_4 = 011$	$s_8 = 24$	$x_8 = 111$

Table 2.1. A code consists of a set of symbols (e.g. decimal numbers) or messages which are encoded as codewords (e.g. binary numbers). Here, the eight symbols are numbers which increase in steps of three, but, in principle, they could be any eight numbers or any eight entities.

The probability of each output is defined by the probability distribution

$$p(Y) = \{p(y_1), \ldots, p(y_m)\}. \tag{2.25}$$

Each output sequence $\mathbf{y}$ is interpreted as implying the presence of a particular input sequence $\mathbf{x}$. If this interpretation is incorrect (e.g. due to channel noise) then the result is an *error*. For a given channel, the mapping between messages and outputs, and *vice versa*, is provided by a *code*. A code consists of a message, an encoder, and a *decoder*. The decoder converts each output $\mathbf{y}$ to a (possibly incorrect) message. Channel noise induces errors in interpreting outputs; the *error rate* of a code is the number of incorrect inputs associated with that codebook divided by the number of possible inputs.

Channel capacity is the maximum amount of information which can be communicated from a channel's input to its output. Capacity can be measured in terms of the amount of information per symbol, and if a channel communicates n symbols per second then its capacity can be expressed in terms of information per second (e.g. bits/s). The capacity of a channel is somewhat like the capacity of a bucket, and the rate is like the amount of water we pour into the bucket. The amount of water (rate) we pour (transmit) into the bucket is up to us, and the bucket can hold (communicate, or transmit reliably) less than its capacity, but it cannot hold more.

In order to be totally clear on this point, we need a few more details. Consider an alphabet of α symbols, where $\alpha = 2$ if the data is binary. If a noiseless channel transmits data at a fixed rate of n symbols/s then it transmits information at a maximum *rate* or channel capacity of $n \log \alpha$ bits/s, which equals n bits/s for binary data.

However, the capacity of a channel is different from the rate at which information is actually communicated through that channel. The rate is the number of bits of information communicated per second, which depends on the code used to transmit data. The rate of a given code may be less than the capacity of a channel, but it cannot be greater; the channel capacity is the maximum rate that can be achieved when considered over all possible codes. For example, a code for binary data

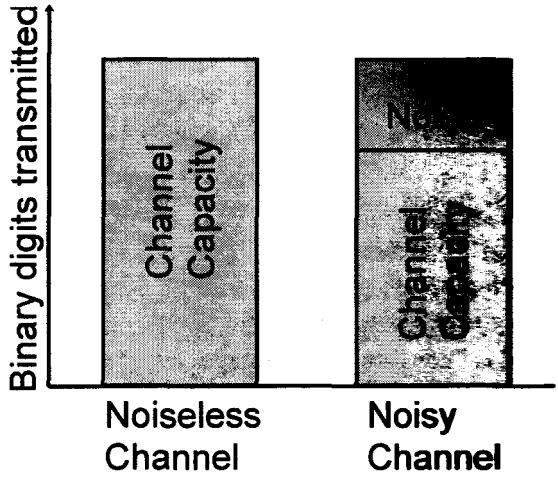


Figure 2.4. The *channel capacity* of noiseless and noisy channels is the maximum rate at which information can be communicated. If a noiseless channel communicates data at 10 binary digits/s then its capacity is $C = 10$ bits/s. The capacity of a noiseless channel is numerically equal to the rate at which it communicates binary digits, whereas the capacity of a noisy channel is less than this because it is limited by the amount of noise in the channel.

in which 0s and 1s occur equally often ensures that each binary digit (symbol) conveys one bit of information, but for any other code each binary digit conveys less than one bit (see Section 2.4). Thus, the capacity of a noiseless binary channel is numerically equal to the rate at which it transmits binary digits, whereas the capacity of a noisy binary channel is less than this, as shown in Figure 2.4.

Some of these definitions require a different interpretation for continuous variables, and we may sometimes use non-bold letters to represent messages, encoded messages and output sequences.

Key point. A message comprising symbols $\mathbf{s} = (s_1, \ldots, s_k)$ is encoded by a function $\mathbf{x} = g(\mathbf{s})$ into a sequence of codewords $\mathbf{x} = (x_1, \ldots, x_n)$, where the number of symbols and codewords are not necessarily equal. These codewords are transmitted through a communication channel to produce outputs $\mathbf{y} = (y_1, \ldots, y_n)$ which are decoded to recover the message $\mathbf{s}$.

2.3. Shannon's Desiderata

Now that we have a little experience of information, we can consider why it is defined as it is. Shannon knew that in order for a mathematical definition of information to be useful it had to have a particular minimal set of properties:

1. **Continuity**. The amount of information associated with an outcome (e.g. a coin flip) increases or decreases continuously (i.e. smoothly) as the probability of that outcome changes.
2. **Symmetry**. The amount of information associated with a sequence of outcomes does not depend on the order in which those outcomes occur.
3. **Maximal Value**. The amount of information associated with a set of outcomes cannot be increased if those outcomes are already equally probable.
4. **Additive**. The information associated with a set of outcomes is obtained by adding the information of individual outcomes.

Shannon[50] proved that the definition of information given below is the only one which possesses all of these properties.

2.4. Information, Surprise and Entropy

Suppose we are given a coin, and we are told that it lands heads up 90% of the time, as in Figure 2.2b. When this coin is flipped, we expect it to land heads up, so when it does so we are less surprised than when it lands tails up. The more improbable a particular outcome is, the more surprised we are to observe it.

One way to express this might be to define the amount of surprise of an outcome value x to be $1/(\text{the probability of } x)$ or $1/p(x)$, so that the amount of surprise associated with the outcome value x increases as the probability of x decreases. However, in order to satisfy the additivity condition above, Shannon showed that it is better to define surprise as the logarithm of $1/p(x)$, as shown in Figure 2.5. This is known as the *Shannon information* of x. (A reminder of the logarithmic function is provided in Appendix C.) The Shannon information of an outcome is also called *surprisal* because it reflects the amount of surprise when that outcome is observed.

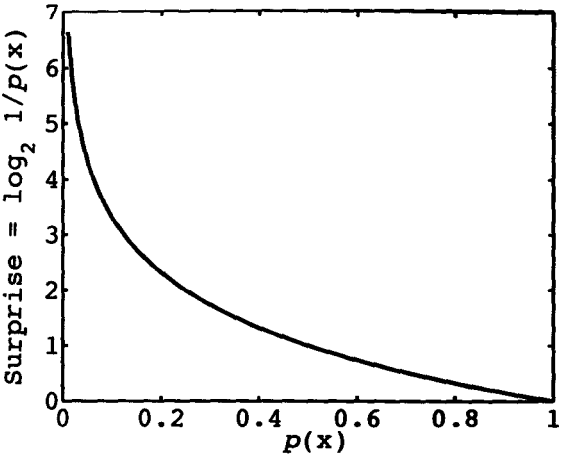


Figure 2.5. Shannon information as surprise. Values of x that are less probable have larger values of surprise, defined as $h(x) = \log_2(1/p(x))$.

If we use logarithms to the base 2 then the Shannon information of a particular outcome is measured in bits,

$$h(x) = \log_2 \frac{1}{p(x)} \text{ bits}, \tag{2.26}$$

where h is standard notation for Shannon information. A general rule for logarithms states that

$$\log_2 \frac{1}{p(x)} = -\log_2 p(x), \tag{2.27}$$

so that Equation 2.26 is often written as

$$h(x) = -\log_2 p(x) \text{ bits}. \tag{2.28}$$

We will usually omit the $_2$ subscript from $\log_2$ unless the base of the logarithm needs to be made explicit.

Key Point. Shannon information is a measure of surprise.

How Surprised Should We Be?

In order to be surprised, we must know which outcomes are more surprising and which are less surprising. In other words, we need to know the probability of the possible outcomes which collectively define the probability distribution $p(X)$ of the random variable X. Thus, the Shannon information implicit in a given set of outcomes can only be evaluated if we know the probability of each outcome. One way to obtain this knowledge is to observe outcomes over a long period of time. Using the observed outcomes, we can estimate the probability of each outcome, and therefore build up an estimate of $p(X)$. But however it is acquired, we need the probability distribution $p(X)$ to evaluate the Shannon information of each outcome.

Entropy is Average Shannon Information

In practice, we are not usually interested in the surprise of a particular value of a random variable, but we would like to know how much surprise, on average, is associated with the entire set of possible values. That is, we would like to know the average surprise defined by the probability distribution of a random variable. The average surprise of a variable X which has a distribution $p(X)$ is called the *entropy* of $p(X)$, and is represented as $H(X)$. For convenience, we often speak of the entropy of the variable X, even though, strictly speaking, entropy refers to the distribution $p(X)$ of X.

Before we consider entropy formally, bear in mind that it is just the average Shannon information. For example, if we flip a coin n times to produce the sequence of outcomes $(x_1, \ldots, x_n)$ then the entropy of the coin is approximately

$$H(X) \approx \frac{1}{n} \sum_{i=1}^{n} \log \frac{1}{p(x_i)}. \tag{2.29}$$

In order to explore the idea of entropy, we will consider examples using two coins: a fair coin, and a coin which lands heads up 90% of the time.

The Entropy of a Fair Coin

The average amount of surprise, unpredictability, or uncertainty about the possible outcomes of a coin flip can be found as follows. If a coin is fair or unbiased then $p(x_h) = 0.5$, as shown in Figure 2.2a, and the surprise of observing a head is

$$h(x_h) = \log \frac{1}{p(x_h)} \tag{2.30}$$

$$= \log(1/0.5) \tag{2.31}$$

$$= 1 \text{ bit.} \tag{2.32}$$

Given that $p(x_t) = 0.5$, the surprise of observing a tail is also one bit. It may seem obvious that the average surprise of this coin is also one bit, but we will make use of some seemingly tortuous reasoning to arrive at this conclusion, because we will require the same reasoning for less obvious cases.

We can find the average surprise by flipping the coin, say, 100 times, measuring the surprise of each outcome, and then taking an average over the set of 100 outcomes. Because each outcome has no effect on any other outcome, these outcomes are *independent.* If we flip a coin

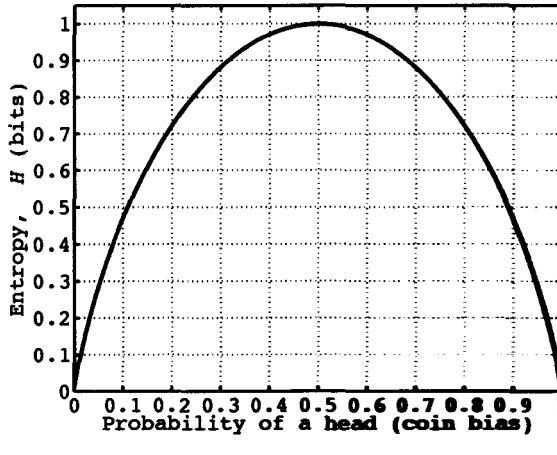


Figure 2.6. Graph of entropy $H(X)$ versus coin bias (probability $p(x_h)$ of a head). The entropy of a coin is the average amount of surprise or Shannon information in the distribution of possible outcomes (i.e. heads and tails), and has a value of one bit for a coin with a bias of $p(x_h) = 0.5$.

100 times we would expect to observe a head roughly 50 times, and a tail roughly 50 times. For the sake of argument, let's assume that we observe exactly 50 heads and 50 tails, so that the average amount of surprise is

$$H(X) = \frac{\left[\sum_{j=1}^{50} \log(1/p(x_h))\right] + \left[\sum_{j=1}^{50} \log(1/p(x_t))\right]}{100} \quad (2.33)$$

$$= \frac{[50 \times \log(1/p(x_h))] + [50 \times \log(1/p(x_t))]}{100}, \quad (2.34)$$

which evaluates to

$$H(X) = [0.5 \times \log(1/0.5)] + [0.5 \times \log(1/0.5)] \quad (2.35)$$

$$= 1 \text{ bit per coin flip.} \quad (2.36)$$

In summary, because the amount of surprise or Shannon information provided by observing the outcome of each flip of this fair coin is one bit, it follows that the average information $H(X)$ of each flip is also one bit.

Interpreting Entropy

There are several ways to interpret the notion of entropy, but one stands out as being particularly accessible. In general, the entropy of a variable is the logarithm of the number m of equally probable outcome values

$$H(X) = \log m \text{ bits.} \quad (2.37)$$

In the above example, there are $m = 2$ equally probable outcome values, so the entropy is confirmed to be

$$H(X) = \log 2 \quad (2.38)$$

$$= 1 \text{ bit.} \quad (2.39)$$

Given that we are using logarithms with a base of 2, we can raise both sides of Equation 2.37 to the power of 2 to confirm that we

are dealing with

$$m = 2^{H(X)} \tag{2.40}$$

$$= 2 \text{ equally probable values.} \tag{2.41}$$

Thus, the number 2 raised to the power of the entropy $H(X)$ yields the number of equally probable outcome values which could be represented by a variable with an entropy of $H(X)$.

Key Point. A variable with an entropy of $H(X)$ bits provides enough Shannon information to choose between $m = 2^{H(X)}$ equally probable alternatives.

The Entropy of an Unfair (Biased) Coin

If a coin is biased then the average amount of surprise or uncertainty is less than that of an unbiased coin, as shown in Figure 2.6. For example, if we know that the probability of a head is $p(x_h) = 0.9$ then it is quite easy to predict the result of each coin flip with a reasonable degree of accuracy (90% accuracy if we predict a head for each flip). If the outcome is a head then the amount of Shannon information is

$$h(x_h) = \log(1/0.9) \tag{2.42}$$

$$= 0.15 \text{ bits per head.} \tag{2.43}$$

On the other hand, if the outcome is a tail then the amount of Shannon information is

$$h(x_t) = \log(1/0.1) \tag{2.44}$$

$$= 3.32 \text{ bits per tail} \tag{2.45}$$

Notice that more information is associated with the more surprising outcome (a tail, in this case).

We will follow the same line of reasoning used for Equation 2.36 to find the average amount of surprise for a coin with bias $p(x_h) = 0.9$.

If we flip this coin 100 times then we would expect to observe a head 90 times and a tail 10 times. It follows that the average amount of surprise is

$$H(X) = \frac{1}{100}\left[\sum_{j=1}^{90}\log\frac{1}{p(x_h)} + \sum_{j=1}^{10}\log\frac{1}{p(x_t)}\right] \quad (2.46)$$

$$= \frac{1}{100}[90 \times \log(1/p(x_h)) + 10 \times \log(1/p(x_t))]. \quad (2.47)$$

Substituting $p(x_h) = 0.9$ and $p(x_t) = 0.1$ yields

$$H(X) = [0.9 \times \log(1/0.9)] + [0.1 \times \log(1/0.1)] \quad (2.48)$$

$$= 0.469 \text{ bits per coin flip.} \quad (2.49)$$

The average uncertainty of this biased coin is less than that of an unbiased coin, even though the uncertainty of one of the outcomes (a tail) is greater for a biased coin (3.32 bits) than it is for an unbiased coin (1 bit). In fact, no biased coin can have an average uncertainty greater than that of an unbiased coin (see Figure 2.6).

Because $p(x_h) = 0.9$ and $p(x_t) = 0.1$, Equation 2.48 can also be written as

$$H(X) = p(x_h)\log(1/p(x_h)) + p(x_t)\log(1/p(x_t)) \quad (2.50)$$

$$= 0.469 \text{ bits per coin flip.} \quad (2.51)$$

If we define a tail as $x_1 = x_t$ and a head as $x_2 = x_h$ then we can write this more succinctly by summing over the two possible values of x_i to obtain the same answer as above:

$$H(X) = \sum_{i=1}^{2} p(x_i)\log\frac{1}{p(x_i)} \quad (2.52)$$

$$= 0.469 \text{ bits per coin flip.} \quad (2.53)$$

As we will see later, an entropy of 0.469 bits implies that we could represent the information implicit in, say, 1,000 flips (which yield 1,000 outcomes) using as little as 469(= 1000 × 0.469) binary digits.

In this example, given that $H(X) = 0.469$ bits, the variable X could be used to represent

$$\begin{aligned} m &= 2^{H(X)} && (2.54)\\ &= 2^{0.469} && (2.55)\\ &= 1.38 \text{ equally probable values.} && (2.56) \end{aligned}$$

At first sight, this seems like an odd result. Of course, we already know that $H(X) = 0.469$ bits is the entropy of an unfair coin with a bias of 0.9. Nevertheless, translating entropy into an equivalent number of equally probable values serves as an intuitive guide for the amount of information represented by a variable. One way to think of this is that a coin with an entropy of $H(X) = 0.469$ bits has the same entropy as an imaginary die with 1.38 sides.

Entropy: A Summary

A random variable X with a probability distribution

$$p(X) = \{p(x_1), \ldots, p(x_m)\} \quad (2.57)$$

has an average surprise (Shannon information), which is its entropy

$$H(X) = \sum_{i=1}^{m} p(x_i) \log \frac{1}{p(x_i)}. \quad (2.58)$$

A succinct representation of this is

$$H(X) = \mathrm{E}[\log(1/p(x))] \text{ bits}, \quad (2.59)$$

where E is standard notation for the **average** or ***expected value*** (see Appendix E).

2.5. Evaluating Entropy

Here, we show that we can **either calculate the entropy of** a variable X from the probability of m **different outcome values** defined by a probability distribution, **or estimate entropy from** n outcome values sampled from that **probability distribution, and that** we obtain

approximately the same value for entropy in both cases. Specifically, as our sample size n grows infinitely large (i.e. $n \to \infty$), the value of the estimate based on the sample converges to the entropy $H(X)$ calculated from the underlying probability distribution. Given that entropy is the average Shannon information of a variable, readers unfamiliar with such matters should read the brief tutorial in Appendix E on how to obtain an average (e.g. entropy) from a distribution.

Calculating Entropy From a Probability Distribution

If we have a die with eight sides, as in Figure 2.7a, then there are $m = 8$ possible outcomes,

$$A_x = \{1, 2, 3, 4, 5, 6, 7, 8\}. \tag{2.60}$$

Because this is a fair die, all eight outcomes occur with the same probability of $p(x) = 1/8$, which defines the uniform probability distribution

$$p(X) = \{1/8, 1/8, 1/8, 1/8, 1/8, 1/8, 1/8, 1/8\} \tag{2.61}$$

shown in Figure 2.7b. The entropy of this distribution can be evaluated using the definition in Equation 2.58, i.e.

$$H(X) = \sum_{i=1}^{m=8} 1/8 \times \log \frac{1}{1/8} \tag{2.62}$$

$$= \log 8 \tag{2.63}$$

$$= 3 \text{ bits.} \tag{2.64}$$

Because the information associated with each outcome is exactly three bits, the average is also three bits and is therefore the entropy of X.

Given that X has an entropy of $H(X) = 3$ bits, it can represent

$$m = 2^{H(X)} \tag{2.65}$$

$$= 2^3 \tag{2.66}$$

$$= 8 \text{ equally probable values.} \tag{2.67}$$

More generally, given a die with m sides, the probability of each outcome is $p(x_i) = 1/m$, so, according to Equation 2.58, the entropy of an m-sided die is

$$H(X) = \sum_{i=1}^{m} 1/m \times \log \frac{1}{1/m} \tag{2.68}$$

$$= \log m \text{ bits}, \tag{2.69}$$

which confirms that entropy is the logarithm of the number of equally probable outcomes.

We can also work back from the entropy $H(X)$ to the value of $p(x_i)$. Substituting $p(x_i) = 1/m$ in 2.65 yields

$$2^{H(X)} = 1/p(x_i), \tag{2.70}$$

so if a die has an entropy of $H(X)$ bits then the probability of each outcome is given by

$$p(x_i) = 2^{-H(X)}. \tag{2.71}$$

Estimating Entropy From a Sample

If we throw a die $n = 1,000$ times then we have a **sample of n outcomes** $\mathbf{x} = (x_1, x_2, \ldots, x_n)$, where each outcome is **chosen from m different** possible outcomes. Given that the Shannon **information of one** outcome x_j is

$$h(x_j) = \log 1/p(x_j), \tag{2.72}$$

we can denote the average Shannon **information of the finite** sample $\mathbf{x}$ as $\overline{h}(\mathbf{x})$, which is calculated as

$$\overline{h}(\mathbf{x}) = \frac{1}{n} \sum_{j=1}^{n} h(x_j) \text{ bits} \tag{2.73}$$

$$= \frac{1}{n} \sum_{j=1}^{n} \log \frac{1}{p(x_j)} \text{ bits} \tag{2.74}$$

If the sample size n is large then the entropy of the sample is approximately the same as the entropy of the random variable X, i.e.

$$\overline{h}(\mathbf{x}) \approx H(X). \tag{2.75}$$

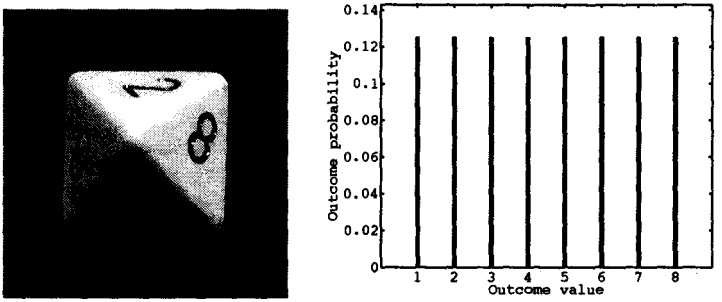
Equation 2.74 is defined over n *instances* of x in a given sample, in which two instances (outcomes) may have the same value, whereas Equation 2.58 is defined over the m *unique outcomes* in X. Usually $n >> m$ (i.e. n is much greater than m).

The definition of entropy in Equation 2.58 looks very different from Equation 2.74, which is more obviously the average Shannon information of a sample of outcome values. This is because Equation 2.58 is used to calculate entropy from a probability distribution of m different outcome values, whereas Equation 2.74 is used to estimate entropy from a sample of n outcome values chosen independently from that distribution.

Whether we calculate entropy from the known probability of each value as in Equation 2.58, or from the average of a sample of values as in Equation 2.74, it is important to note that entropy is the average amount of Shannon information provided by a single value of a variable.

2.6. Properties of Entropy

In essence, entropy is a measure of *uncertainty*. When our uncertainty is reduced, we gain information, so information and entropy are two sides of the same coin. However, information as conceived by Shannon



(a) (b)

Figure 2.7. (a) An 8-sided die. (b) The uniform normalised histogram (probability distribution) of outcomes has an entropy of $\log 8 = 3$ bits.

has a rather subtle interpretation, which can easily lead to confusion. Accordingly, it is worth a moment's thought to consider the meaning of information and entropy.

Average information actually shares the same definition as entropy, but whether we call a given quantity information or entropy usually depends on whether it is being given to us or taken away. For example, a variable may have high entropy, so our initial uncertainty about the value of that variable is large and is, by definition, exactly equal to its entropy. If we are then told the value of that variable then, on average, we have been given an amount of information equal to the uncertainty (entropy) we initially had about its value. Thus, receiving an amount of information is equivalent to having exactly the same amount of entropy taken away.

Key point. The average uncertainty of a variable X is summarised by its entropy $H(X)$. If we are told the value of X then the amount of information we have been given is, on average, exactly equal to its entropy.

Doubling the Number of Sides

If we double the number of sides on the die from eight to 16 then we double the number of possible outcomes. Following the same line of reasoning as in Equations 2.62–2.64, the entropy of outcome values is $H(X) = \log 16 = 4$ bits. Doubling the number of sides from eight to 16 increases the entropy from three to four bits, and therefore adds exactly one bit (i.e. $\log 2$) to the entropy of the distribution of outcome values.

Key point. Doubling the number of possible values of a variable adds one bit to its entropy.

Doubling the Number on Each Side

As we have just seen, the entropy of a discrete random variable X depends on the *number* of different outcome values and on the probability of each outcome value. But it does not depend on the particular outcome values that X can adopt. For example, we could

double all the outcome values on an 8-sided die to define the alphabet $A_x = \{2, 4, 6, 8, 10, 12, 14, 16\}$. However, each of these values would still occur with the probability (1/8) defined in Equation 2.61, so the entropy would still be $H(X) = 3$ bits. In contrast, doubling the values of a continuous variable does change its entropy (see Section 5.5).

Key point. The entropy of a discrete variable depends only on the probability distribution of its values. Changing the values of a discrete variable does not change its entropy provided the *number* of values stays the same.

2.7. Independent and Identically Distributed Values

When a die is thrown, each outcome value is chosen from the same uniform distribution of values. Additionally, it does not depend on any other outcome, so values are said to be *independent.* Outcome values that are chosen independently from the same uniform distribution are said to be *independent and identically distributed*, which is usually abbreviated to *iid.*

These considerations imply that the entropy of a sequence is given by Equation 2.58 only if its values are chosen independently from the same distribution (i.e. if they are iid). However, if consecutive values are related (e.g. as in an English sentence) then they do not provide independent information. In this case, the elements of the sequence are not iid, and the sequence has less entropy than the summed (over-estimated) entropies of its individual elements calculated using Equation 2.58.

If a source generates values chosen from an underlying distribution which remains constant over time then the source is said to be *stationary.* Unless stated otherwise, all sources in this text are assumed to be stationary.

2.8. Bits, Shannons, and Bans

The maximum amount of information associated with a discrete variable is the logarithm of the number m of equally probable values it can adopt. Thus, because a binary variable can adopt $m = 2$ states, it conveys up to $n = 1$ bit of information.

However, the translation from m possible values to an amount of information depends on the base of the logarithm used to do the translation (see Appendix C). If we used base 10 instead of base 2 then we would obtain a different answer, even though the underlying amount of information would remain the same. By analogy, consider an amount of water that can be expressed either as two pints or about 1.14 litres; the amount of water is the same in both cases, but the units in which it is measured are different.

If we measure information using logarithms with the base $e = 2.72$ (as used in natural logarithms) then the units are called *nats*. In contrast, if we measure information using logarithms with base 10 then the units are called *bans*, named after the English town of Banbury. The ban was named during World War II by the British code-breakers of Bletchley Park (including Alan Turing), where the German code had to be broken on a daily basis. Data were tabulated using special cards or *banburies* printed in Banbury, and the code-breaking method was called *Banbarismus*[34]. Finally, because the word bit is often mistakenly used to refer to a binary digit (see Section 1.5), a less ambiguous name is the *Shannon*[34], for which an appropriate abbreviation would be *Sh*.

2.9. Summary

Entropy is a core concept in information theory. But it is also quite subtle and demands a sound grasp of probability, random variables and probability distributions, which were introduced in this chapter. After defining key technical terms, we considered entropy as the average amount of surprise of a particular variable, like the flipping of a coin or the throw of a die. Both of these were used as examples for calculating entropy.

Chapter 9

Information As Nature's Currency

> Nature uses only the longest threads to weave her patterns, so each small piece of her fabric reveals the organization of the entire tapestry.
> Feynman R, 1967.

9.1. Introduction

Information theory is used in research fields as diverse as linguistics, communications, signal processing, computing, neuroscience, genetics, and evolutionary theory. In this chapter, we explore some relevant applications, and discuss them in the context of information theory.

9.2. Satellite TVs, MP3 and All That

The next time you switch on your TV, remember that what you see on the screen bears only a superficial resemblance to the data streaming out of the satellite that delivers the signal to your house. The reason is that the original images were encoded or *compressed* before being transmitted from a TV station to the satellite, and these encoded images must be decoded by your TV's computer before the original images can be recovered. Thus, the problem faced by modern television systems is that images must be displayed at a very high quality, but the image data must be transmitted through a communication channel with a relatively small capacity. This is clearly a problem that can be addressed with information theory.

The camera that recorded the original image data is essentially an information source that generates data at a rate of about 1.5 gigabits per second (Gb/s), where a gigabit is one billion (10^9 or 1,000 million)

binary digits. This figure of 1.5 Gb/s results from the fact that TVs typically display 1920 elements horizontally and 1080 lines vertically, at a rate of 30 images per second. Each colour image displayed consists of 1920 × 1080 sets of three pixels (red, green and blue), so the total number of pixels is about 6 million (i.e. $3 \times 1920 \times 1080 = 6{,}220{,}800$).

Let's assume the intensity of each pixel has a range of zero to 255, which is represented by $\log 256 = 8$ binary digits, making a total of 49,766,400, or about 50 million binary digits per image. Because a TV displays 30 images per second, this amounts to about 1,500 million binary digits/s. This is confusingly quoted as 1,500 megabits/s (i.e. 1,500 million *bits*/s) in the world of computing because both bit and binary digit are used to refer to binary digits (see Section 1.5).

However, the channel through which these 1,500 million binary digits/s are communicated (i.e. a satellite) can carry only 19.2 million binary digits/s. According to Shannon, a noiseless channel which carries 19.2 million binary digits/s has a capacity of exactly 19.2 million bits/s. So the problem is this: how can the information implicit in 1,500 million binary digits/s be communicated through a channel which carries only 19.2 million binary digits/s?

Roughly speaking, the solution comprises four stages of processing before the signal is transmitted. First, squeeze all of the redundant data out of the images. Second, remove components which are essentially invisible to the human eye. Third, recode the resultant data so that all symbols occur equally often (e.g. using Huffman coding), and, finally, add a small amount of redundancy in the form of an error-correcting code. As discussed in Chapter 1, the similarity between adjacent pixel values is one source of redundancy. However, because a TV receives data as a temporal sequence of images, the state of each pixel tends to be similar between consecutive images, and this temporal redundancy can also be removed to compress the data.

The standard set of methods used to remove spatial and temporal redundancy are collectively called MPEG (Moving Picture Expert Group). Whilst these methods are quite complex, they all rely heavily on a core method called the *cosine transform*. In essence, this decomposes the data into image features of different sizes. When

measured across space (i.e. within an image), size translates to *spatial frequency* (actually, 1/spatial frequency, so large features have low spatial frequencies). When measured across time, size translates to features that persist for short or long times, which translates to *temporal frequency* (actually, 1/temporal frequency, so persistent features have low temporal frequencies).

Once the data have been transformed into a series of spatial and temporal frequencies, some of them can be thrown away. This is an important step for two reasons. First, it reduces the quantity of data to be communicated, which is good. Second, it reduces the quantity of information communicated, which is bad. However, if the information thrown away involves only those frequencies which we cannot see, then it is not so bad, after all. For example, fine details that are beyond the resolution of the human eye correspond to high spatial frequencies, and can therefore be discarded without any apparent loss of image quality. Similarly, changes over time that are too fast to be detectable by the eye correspond to high temporal frequencies, and can be discarded. At the other extreme, very low spatial and temporal frequencies are also essentially invisible to the eye, and can be discarded.

The eye is sensitive to intensity (luminance), but relatively insensitive to fine gradations of colour. Consequently, the data from TV cameras can be recoded to give high-resolution intensity data, and low-resolution colour data. Because this recoding mirrors the encoding within the human visual system[51], the perceived picture quality is unaffected. The result is a 50% saving in the number of binary digits required to represent information regarding the colour and intensity of pixels.

Discarding certain spatial and temporal frequencies and recoding intensity/colour data means that data recorded at a rate of 1,500 million binary digits/s can be compressed and transmitted through a communication satellite channel with a capacity of 19.2 million bits/s, and then decoded to present you (the TV viewer) with 1,500 million binary digits of data per second (with some loss of information but no visible loss of quality). This represents an effective compression factor of about 78($\approx$ 1500/19.2), so it looks as if we can communicate 78 times more data than the channel capacity would suggest. In fact, the

compression has to be a little better than this, because the stereo sound is also squeezed into the same channel. This is achieved by applying the cosine transform mentioned above to sound (where it is called MP3).

Compression that throws away information is called *lossy*, whereas compression that preserves all information is called *lossless*. Most practical systems are lossy, but they do not give the appearance of losing information because the discarded information is invisible, or, at least, unimportant for human perception of images or sounds.

Finally, in order to minimise the effects of noise, redundancy is added to the compressed data. Unlike the redundancy of the original data, the amount of added redundancy is small, but it is just enough to enable the receiver to recover from all but the most catastrophic forms of noise in the form of electrical interference.

9.3. Does Sex Accelerate Evolution?

Even though the Darwin–Wallace theory of evolution by natural selection was published in 1859, it still holds a number of mysteries. Prominent amongst these is the question of sex. Specifically, why do some species have two sexes?

Many answers which have been proposed rely on logical argument mixed with a degree of plausible speculation, rather than mathematical analysis. And even though mathematical analysis cannot usually provide definitive answers to biological questions, it can constrain the space of possible answers. In so doing, some answers can be definitely excluded, and those that remain can be used to yield hypotheses which can be tested empirically.

Evolution is essentially a process in which natural selection acts as a mechanism for transferring information from the environment to the collective genome of a species. (The term genome is conventionally used to refer to all of the genes in a *particular* individual, but we use it to refer to all of the genes in a *typical* individual.) Each individual represents a question asked of the environment: are the genes in this individual better or worse than average? The answer is often brutal, because the environment destroys many individuals in infancy. But even when the answer is given more tactfully, so that an individual does not die but simply has fewer offspring, it is still brutally honest.

The answer comes in units called fitness, and a good measure of fitness is the number of offspring each individual rears to breeding age. Over many generations, the information provided by the answers (fitness) allocated to each individual coerces the genome via natural selection to adopt a particular form. Thus, information about the environment eventually becomes implicit in the genome of a species.

The form of the information received from the environment is both crude and noisy. It is crude inasmuch as it is measured in units of whole numbers of offspring, rather than in the fractional offspring numbers required to obtain a precise measure of fitness. And it is noisy inasmuch as the environment does not simply provide a perfect estimate of each individual's fitness because, sometimes,

> the race is not to the swift, nor the battle to the strong
> ... but time and chance happeneth to them all.
> Ecclesiastes 9:11, King James Bible.

Thus, the environment assigns a noisy fitness value to each individual, where fitness equates to the number of offspring each individual rears to adulthood. However, this fitness value gets assigned to a whole individual, and not to particular genes that make large (negative or positive) contributions to an individual's fitness. In other words, the totality of an individual's fitness value does not specify which beneficial features increase fitness, or which detrimental features decrease fitness. Ultimately, each individual either lives or dies before reproducing, which implies that its genome provides a maximum of one bit of information about its relative fitness. However, the fact that the information received from the environment is crude and noisy is not necessarily a problem for evolution.

In a theoretical *tour de force*, John Holland (1992)[25] proved that the type of genetic algorithm used in natural selection implements a kind of *intrinsic parallelism*, which makes it incredibly efficient at allocating blame or credit to particular genes. In essence, Holland's mathematical analyses proved the *schema theorem*, which states that a gene increases in frequency within a species at a rate exponentially related to the extra fitness that gene confers on its owners (i.e. good genes spread extremely quickly). Even though Holland's results make no explicit claims on

information theory, it seems likely that information theory is relevant. On a related topic, information theory has been applied to test the idea that the genetic code is optimal[33;44].

The human genome contains about 3×10^9 (3 billion) pairs of nucleotides, where each nucleotide comprises one element in one half of the classic double helix of the DNA (deoxyribonucleic acid) molecule (there are about 1,000 nucleotides per gene). In order to explore how quickly the genome acquires information about the environment, MacKay (2003)[34] identifies one bit as corresponding to one nucleotide, which is a well-defined chemical structure (unlike a gene). The human genome makes use of four particular nucleotides, adenine (A), guanine (G), thymine (T) and cytosine (C). Within the DNA molecule, nucleotides pair up on opposite sides of the double helix, such that A pairs with T, and C pairs with G.

In the spirit of the methods applied successfully in physics, MacKay uses a stripped-down model of each nucleotide, which is assumed to occur in one of two states, good or bad. Given a genome of N nucleotides, the fitness of an individual is defined simply as the number of nucleotides in a good state, and the normalised fitness is defined as

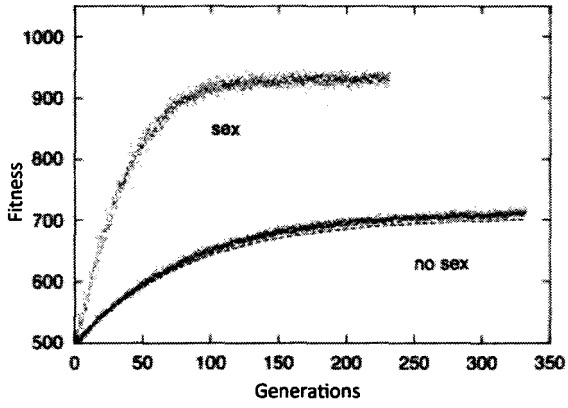


Figure 9.1. How fitness changes over generations for asexual and sexually reproducing populations. In this simulation, the number of genes per individual is $N = 1,000$, and 1,000 individuals initially had randomly generated genomes with a fitness $f = 0.5$, so the population fitness was initially 500. The dashed line shows the theoretical curve. Reproduced with permission from MacKay (2003)[34].

the proportion of good genes in an individual's genome. Finally, the proportion of good genes above 50% is defined as the *excess normalised fitness*, δf, which we will just call *fitness*.

MacKay then compares the effects of different mutation rates on fitness in populations that reproduce sexually and asexually. If a sexually reproducing population is well adapted to its environment then, by definition, the genome is close to an ideal or optimal genome. MacKay shows that this near-optimal genome acquires one bit every two generations. However, if the environment changes then over generations the genome adapts. MacKay proves that the rate at which a genome of N nucleotides accumulates information from the environment can be as large as

$$\sqrt{N} \text{ bits/generation.} \tag{9.1}$$

For example, given that $N = 3 \times 10^9$ nucleotides, the rate at which the population accumulates information from the environment can be as large as 540,000 bits/generation. In this case, the collective genome of the current generation would have 540,000 bits more information about its environment than the genomes of the previous generation, and so should be better prepared to meet the challenges of that environment. In contrast, under similar circumstances, an asexually reproducing population (e.g. aphids) acquires information at a fixed rate of one bit per generation.

The result stated in Equation 9.1 seems to suggest that the bigger the genome, the faster evolution can work. If this is true, then why doesn't the process of natural selection result in genomes which are as large as possible? The answer involves mutation rates.

The mutation rate is the probability that each gene will be altered from one generation to the next, and (equivalently) is the average proportion of genes altered from one generation to the next. It turns out that, in theory, the largest mutation rate that can be tolerated in an asexual population is about $1/N$, whereas it is $1/\sqrt{N}$ in a sexual population. To take a simple example, if the genome size is tiny, say $N = 100$, then the largest mutation rate that can be tolerated in an asexual population is 0.01, whereas it is $1/\sqrt{N} = 1/10 = 0.1$ in a

sexual population. So, in both cases, the rate of genetic mutation affects large genomes more than small ones, but a sexual population can tolerate a much higher mutation rate than an asexual population. This is especially pertinent given that mutation is pretty much the only source of inter-individual variability available to asexual populations. Because variability provides the raw materials for natural selection, the net effect is to accelerate evolution in sexual populations.

However, the above results provide a biological conundrum: a large genome increases evolutionary speed, but it also decreases tolerance to mutations. This means there is a trade-off between a large genome (which accelerates evolution, but provides poor mutation tolerance) and a short genome (which provides good mutation tolerance, but decelerates evolution). These conflicting constraints suggest that evolution should have found a 'happy medium' for genome size, which ensures that the rate of evolution is as fast as it can be for a given genome size and mutation rate.

The upshot of this analysis suggests that natural selection allows a genome of size N to accumulate information from the environment at a rate proportional to $\sqrt{N}$ times faster in a sexual population than in an asexual population. Additionally, a sexual population can tolerate a mutation rate that is proportional to $\sqrt{N}$ times greater than the mutation rate that can be tolerated by an asexual population. For genomes with $N \approx 10^9$ nucleotides, this factor of $\sqrt{N}$ is not trivial. Even for a 'toy' genome size of $N = 10,000$, if an asexual population accumulates information at the rate of 10 bits/generation then a sexual population would accumulate information at the rate of $10 \times \sqrt{10,000} \approx 1,000$ bits/generation.

Darwin would almost certainly approve of this type of analysis, which seeks to find the laws which underpin evolution by natural selection:

> The grand Question which every naturalist ought to have before him when dissecting a whale or classifying a mite, a fungus or an infusorian is "What are the Laws of Life?".
> Darwin C, *B Notebook*, 1837.

According to the analysis summarised above, we now have a candidate for one of these laws: between successive generations, the collective

genome of a species should maximise the Shannon information acquired about its environment for each joule of expended energy. This general idea of *efficient evolution* can be considered to be an extension of the efficient coding hypothesis normally applied to brains.

Finally, MacKay argues that faulty transcription of DNA effectively smoothes out an otherwise rugged fitness landscape, and thereby increases the rate of evolution. This type of phenomenon is an example of the *Baldwin effect*[4], which is traditionally considered to rely on within-lifetime learning to accelerate Darwinian genetic evolution.

The results of information-theoretic analyses[34;54] of biological evolution are not only compelling and intriguing; they also provide hypotheses that can be tested empirically: hypotheses which should make redundant much of the speculative debate that often accompanies such controversies.

9.4. The Human Genome: How Much Information?

Like any message, DNA is composed of a finite alphabet. As described above, the alphabet for DNA comprises 4 letters (A,G,T, and C), where each letter corresponds to one nucleotide. For a message which is N letters long, the number m of possible messages that could be sent is therefore $m = 4^N$, and the maximum amount of information conveyed by this message is $H = \log 4^N = N \log 4 = 2N$ bits. Given that the human genome consists of $N = 3 \times 10^9$ nucleotides, this implies that it contains a maximum of $H = 6 \times 10^9$ or 6 billion bits of information. Of course, the genetic code is partially redundant (which may be necessary to minimise errors) and some regions of the DNA molecule do not seem to code for anything, so the estimate of 6 billion bits is an upper bound.

Note that 6 billion bits could be stored in 6 billion binary digits, which is almost one gigabyte (a gigabyte is 8×10^9 binary digits). For comparison, a basic DVD disc can store 8 billion binary digits, which is enough for a two-hour movie. However, whereas a disc or a memory stick consists of a substantial amount of material, these 6 billion binary digits' worth of data are stored in the DNA inside the nucleus of every cell in the human body.

9.5. Enough DNA to Wire Up a Brain?

Everything that we see, hear, or touch depends on the flow of information through nerve cells or *neurons*. These neurons are the only connection between us and the physical world, and the brief on–off pulses they deliver to the brain are the only messages we can ever receive about that world. However, even before we are born, some information has already been transferred from our DNA into the brain; otherwise, we could neither breathe nor suckle. So at least some of the brain's microstructure is determined primarily by nature, rather than nurture. In terms of information, an obvious question is: does the DNA of the human genome contain enough information to specify every single connection in the brain of a new-born baby?

The human brain contains about 10^{11} (one hundred billion) neurons, and each neuron has about 10^4 (ten thousand) connections or *synapses* to other neurons. This leads to an estimate of 10^{15} synapses in total. As stated above, the human genome contains about 3×10^9 (three billion) nucleotides (see Section 9.3). In fact, there are about 1,000 nucleotides per gene, but let's keep things simple. If each nucleotide specified one synapse then this would place an extremely conservative upper bound on the number of synapses that could be genetically programmed. Even under this conservative assumption, if one bit were used to specify the strength of a synapse (e.g. on or off) then there would be enough information for only 10^9 synapses, which represents one millionth of all the synapses in the brain ($10^9/10^{15} = 10^{-6}$). And this would leave no DNA for the rest of the body. In essence, if we want to use DNA to encode synapses then we would need about a million times more DNA than we have now.

In Darwin's day, the question posed above would be hostage to pure speculation. But with Watson and Crick's discovery of the structure of DNA in 1953, combined with Shannon's information theory, we can answer this question definitively: *no*, there is not enough DNA in the human genome to specify every single connection in the brain.

A compelling implication of this answer is that the human brain must learn. It may seem obvious from observing an infant that we learn, but the fact that we do learn does not imply we *have to* learn in order to

develop. Thus, information theory tells us that we *must* learn, because learning provides a way for the brain to use the information supplied by the environment to specify the correct set of 10^{15} synapses.

9.6. Are Brains Good at Processing Information?

Neurons communicate information. They do not care about the shape of a tree, the smell of vanilla, or the touch of a cat's fur. That is for us, the beneficiaries of what neurons do; what they do is to communicate information, and that is pretty much all that they do.

Ever since the first neurons appeared, about 580 million years ago, the relentless forces of natural selection have ensured that they are about as perfect as they can be. We know this is true because information theory has been applied within the research field of *computational neuroscience* to the various functions that neurons perform[17;35;43]. The success of this highly technical research program, far from diminishing the wonder of brain function, promises a radical change in our perspective:

> ... we claim that the results of a quantitative approach are sufficiently extreme that they begin to alter our qualitative conception of how the nervous system works.
> Rieke *et al*, 1997[43].

A major reason this research program has been so successful is because, whatever else neurons appear to be doing, they must be communicating information about the world to the brain. As the world offers up almost infinite amounts of sensory data to be communicated, it is necessary for neurons to be selective, and efficient in encoding the information they select for transmission. This general approach has been championed by Horace Barlow, who has been largely responsible for the genesis of the resultant *efficient coding hypothesis*[5]. Put simply, this states that data acquired by the eye should be encoded as efficiently as possible before being communicated to the brain.

Information in Spiking Neurons

One of the first applications of information theory to neuroscience was by MacKay and McCulloch (1952), who calculated the entropy of a spiking source, a mere four years after the publication of Shannon's

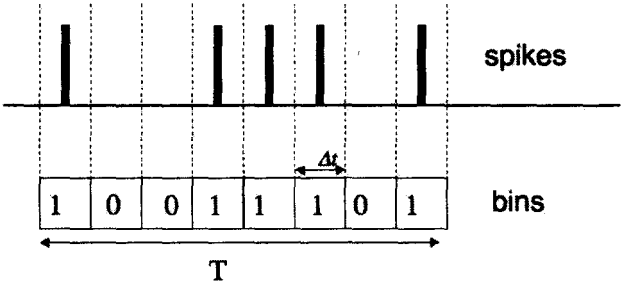


Figure 9.2. A series of n voltage spikes from a neuron can be represented as 0s and 1s. If a time period of T seconds contains n spikes then the firing rate is $r = n/T$ spikes/s. If T is divided into M small intervals or bins of width Δt then the probability of observing a spike in each bin is $p = r \times \Delta t$.

paper. The following account is based on Rieke *et al* (1997)[43], and on lecture notes by Professor John Porrill.

Suppose we regard the neuron as a channel. A neuron acts as a binary channel because neurons deliver information in the form of brief voltage spikes, as shown in Figure 9.2. These spikes occur at a typical rate of between one and 100 spikes per second (spikes/s). Suppose we record spikes with only limited temporal resolution of, say, $\Delta t = 0.001$ seconds (i.e. 1 millisecond (ms)) then a spike can either be present or not present in each 1 ms interval. This binary channel can carry one binary digit each millisecond, so its capacity is $C = 1/\Delta t$ bits/s. With a temporal resolution of $\Delta t = 0.001$ seconds, this gives an upper bound defined by its channel capacity of 1,000 bits/s.

Let's estimate the entropy of the output of a typical neuron, assuming that spikes are mutually independent. The entropy of a sequence of spikes, or *spike train*, is given by the logarithm of the number of different possible spike trains that could occur in a given time interval T. In order to avoid the complicated mathematics that ends with a good approximation to this[43], we will use some much less complicated mathematics to obtain a slightly less good approximation.

If spikes are produced at an average firing rate of r spikes/s then the probability of a spike in each time interval of Δt seconds is

$$p = r\,\Delta t, \tag{9.2}$$

and the probability that no spike occurs in each interval is therefore $q = 1 - p$. Thus, the average Shannon information of each interval is

$$H = h(p) + h(q) \tag{9.3}$$

$$= p\log(1/p) + q\log(1/q) \text{ bits.} \tag{9.4}$$

If we define Δt to be sufficiently small then the probability of a spike in each interval is low, and therefore the probability of no spike is almost one, which implies that $h(q) \approx 0$ bits. Thus, Equation 9.4 can be approximated by its first term $H \approx p\log(1/p)$. We can use Equation 9.2 to rewrite this in terms of Δt

$$H \approx r\,\Delta t \log \frac{1}{r\,\Delta t} \text{ bits.} \tag{9.5}$$

This is actually the average Shannon information produced in time Δt, so the average rate R at which Shannon information is generated is found by dividing by Δt

$$R = H/\Delta t \tag{9.6}$$

$$= r \log \frac{1}{r\,\Delta t} \text{ bits/s,} \tag{9.7}$$

where we have omitted the approximation symbol. For a neuron with a firing rate of $r = 50$ spikes/s, and assuming an interval of one millisecond ($\Delta t = 0.001$s), this gives an information rate of

$$R = 50 \log \frac{1}{50 \times 0.001} = 216 \text{ bits/s.} \tag{9.8}$$

If 50 spikes convey 216 bits then each spike conveys an average of

$$216/50 = 4.32 \text{ bits/spike.} \tag{9.9}$$

Note that the temporal precision Δt effectively places an upper bound on channel capacity. This bound applies to us (as observers of neuronal outputs) and to neurons that receive the outputs of other neurons which cannot resolve the timing of spikes below some threshold of temporal precision.

The estimate of 4.32 bits/spike is more than the information (one bit) carried by the state (spike or no-spike) of each bin, which can be understood if we adopt a slightly different perspective. Given a firing rate of 50 spikes per second and a temporal resolution defined by 1,000 bins per second, an average of one out of every 20 (=1000/50) bins contains a spike. Because each spike can appear in any bin with equal probability, each set of 20 bins can adopt 20 equally probable states. Roughly speaking, each set of 20 bins has an entropy of about $\log 20 = 4.32$ bits, and because each of these sets contains an average of one spike, each spike effectively conveys an average of 4.32 bits.

Irrespective of precisely how much information is implicit in the neuron's output, there is no guarantee that it provides *information about the neuron's input*, that is, the *mutual information* between the input and output. Note that the mutual information cannot be greater than the maximum entropy implied by the neuron's mean firing rate. Thus, the maximum entropy implied by a neuron's mean firing rate acts as an upper bound on its entropy, and this, in turn, acts as an upper bound on the mutual information between the neuron's input and output.

Using data collected from mechanical receptors in the cricket, Warland *et al* (1992) found that neurons have an entropy of about 600 bits/s. However, it was found that only about half of this entropy is related to the neuron's input, and the rest is noise. These neurons therefore transmit information about their inputs at a rate of about 300 bits/s, which represents a *coding efficiency* of about 0.5 (i.e. 300/600).

Let's think about what it means for a neuron to provide 300 bits/s about its input. Using a simplistic interpretation, at the end of one second a neuron has provided 300 bits, which is enough information to specify its input to within one part in 2^{300}, or (equivalently) as one part in 2×10^{90}. This would be analogous to measuring someone's height to within a fraction of the width of an atom, which is clearly silly. So what are we to make of this result?

An alternative interpretation, proposed by Rieke *et al*[43], is that each neuron provides a kind of 'running commentary' about its input, which is usually changing rapidly. When considered like this, each neuron

provides, on average, three bits each 10 ms, where 10 ms seems to represent a plausible time frame. In other words, every 10 ms a neuron's output specifies what is driving its input with a precision of one part in $8(= 2^3)$. For example, if a neuron is sensitive to the speed of visual motion between, say, zero and 32 degrees/s then its output over 10 ms could indicate one particular eighth of this range. Because 1/8 of 32 is 4, the neuron's output could indicate a speed of, say, 20 ± 2 degrees/s, where ± 2 implies a range of 4 degrees/s. In this case, the information conveyed over 10 ms is about the same as the information conveyed by each spike (three bits), so the timing of each spike effectively indicates speed to within ± 2 degrees/s.

Experiments on the frog auditory system[42] showed that the proportion of output entropy that provides information about a neuron's input depends on the nature of the sounds used. Specifically, if artificial sounds were used then the coding efficiency was only about 20%, but if naturalistic frog calls were used then the coding efficiency was an impressive 90%.

Eyes, Flies, and Information Theory

The purpose of an eye is to communicate information about the world to the brain, and this is as true for the eye of a fly as it is for the eye of a human. Thus, even though the structure of a human eye is very different from the structure of a fly's eye, it is likely that the underlying computational processes in both are essentially the same.

Laughlin (1981)[31] showed that neurons which receive outputs from the eye communicate about as much information as is theoretically possible. He showed this to be true for fly eyes, and subsequent work suggests that it is also true for human eyes. Laughlin's brief paper (two pages) is not only a physical example of the power of information compression, but also one of the most insightful papers in biology.

Organisms with eyes are more interested in differences in luminance, or *contrast*, than in luminance *per se*. For this reason, the neurons which receive outputs from photoreceptors respond to contrast rather than luminance. In the fly, these neurons are called *large monopolar cells* or LMCs. These brain cells, which have continuous voltage

outputs (rather than spikes), are the cells from which Laughlin made his recordings.

The first question Laughlin posed in his paper was this: if the distribution of contrasts seen by a fly's eye is fixed by its natural environment, and if the LMC neurons in a fly's eye have a specific

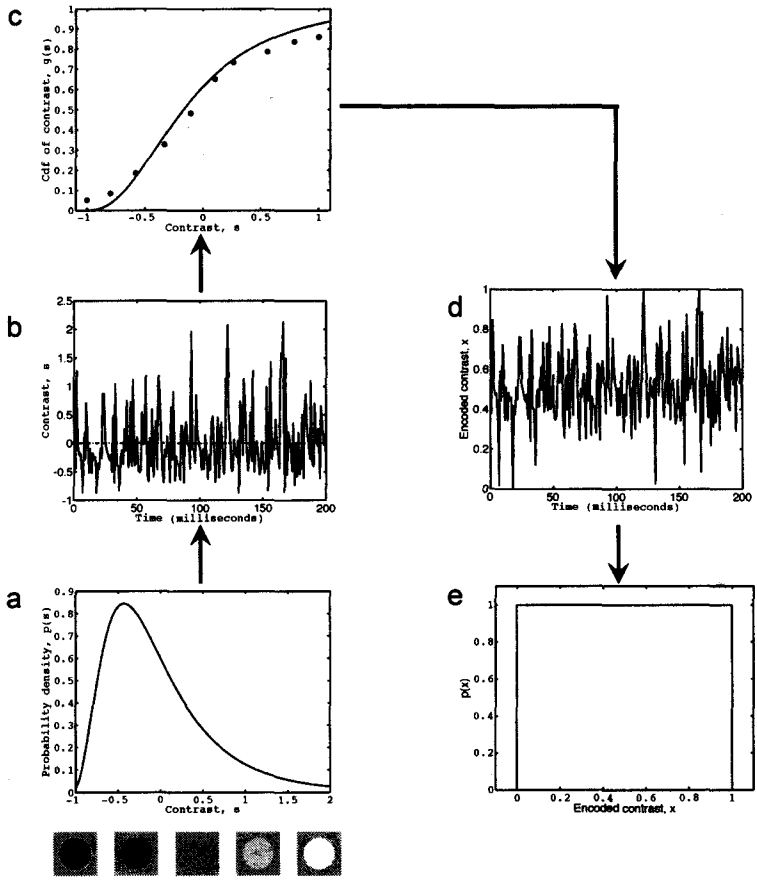


Figure 9.3. Schematic summary of Laughlin's experiment.
(a) Probability density function $p_s(s)$ of contrasts s in the fly's environment.
(b) Sampled values of s from (a) obtained over time as the fly moves around.
(c) Transforming s values to x values using the cumulative distribution function of $p_s(s)$, which is the optimal encoding function g^* (smooth curve), and which predicts the (rescaled) outputs $x = g(s)$ of LMC neurons (dots).
(d) How LMC neuron outputs x change over time.
(e) Uniform pdf $p_x(x)$ of LMC outputs predicted by g^*.

input/output encoding function, then what is the precise form of the encoding function which would communicate as much information as possible from the fly's eye to the fly's brain?

Each LMC neuron has an output between $x_{min} = -20\,\text{mV}$ and $x_{max} = +20\,\text{mV}$, so it acts like an encoding function which prepares (encodes) contrasts for transmission along a communication channel with fixed lower and upper bounds (where LMC outputs were rescaled to have a range between zero and one). The analysis in Section 7.7 implies that the *optimal encoding function* $g^*(s)$ for such a channel is given by the cumulative distribution function (cdf) of the probability density function (pdf) $p_s(s)$ of contrasts s in the fly's environment $x^* = g^*(s)$. By definition, if the pdf of contrasts seen by a fly's eye in its natural environment is $p_s(s)$ then the cdf of contrast values is

$$g^*(s) = \int_{t=-\infty}^{s} p_s(t)\,dt. \tag{9.10}$$

From Section 7.7, we know that the encoding function g^* is guaranteed to transform the distribution $p_s(s)$ of contrast values into a uniform pdf $p_x(x)$ of encoded contrast values x. Because the uniform distribution is a maximum entropy pdf, each value of x provides as much information as possible.

In order to estimate the optimal encoding function $g^*(s)$, Laughlin needed to know $p_s(s)$, the pdf of contrasts in the fly's environment. For this, he measured the distribution of contrasts which occur in a woodland setting. These data were used to construct a histogram, which represents an approximation to $p_s(s)$ (Figure 9.3a). Numerical integration of this histogram was then used to estimate the cdf $g^*(s)$ (Figure 9.3c, solid curve).

Having used information theory to find the precise form that an optimal encoding function should adopt, Laughlin's second question was this: does the *LMC encoding function* g implicit in LMC neurons match the optimal encoding function g^* predicted by information theory?

In some respects, this second question is conceptually more straightforward than the first question because the encoding function g

of an LMC neuron is given by the mean response to each contrast. Accordingly, Laughlin exposed the fly to different contrasts in his laboratory, and measured the output voltage of LMC neurons. As the contrast was increased, the output increased slowly at first, then more rapidly, and finally tailed off at very high contrasts, as shown by the data points plotted in Figure 9.3c.

To answer his second question, Laughlin had to compare the data points from the neuron's encoding function $g(s)$ with the optimal encoding function $g^*(s)$. In order to make this comparison, LMC outputs were linearly rescaled to have a range between zero and one. The match between the LMC outputs $x = g(s)$ (the data points in Figure 9.3c) and the values predicted by the optimal encoding function $g^*(s)$ (the solid curve in Figure 9.3c) is remarkably good.

Laughlin's experiment represents one of the first tests of an information-theoretic optimality principle within the brain (i.e. the efficient coding hypothesis). This general approach has been vindicated in tests on other organisms (including humans[38;51;55]) and on other sense modalities (e.g. olfaction[29] and audition[42]). Indeed, Karl Friston's *free-energy theory*[17;47] assumes that an organising principle for all behaviour consists of minimising the sum total of all future surprises (i.e. sensory entropy).

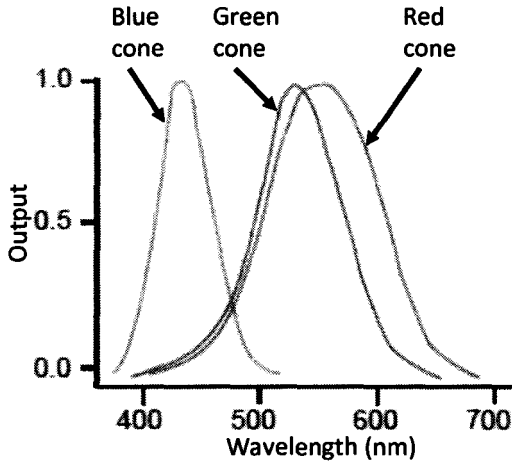


Figure 9.4. Photoreceptor sensitivity to different wavelengths of light.

The Colour of Information

The human eye contains about 126 million photoreceptors, which consist of two main types. The 120 million *rods* are very sensitive to light, and these function only in low-light conditions. The 6 million *cones* function under daylight conditions, and are concentrated close to the optical centre of the eye. There are three types of cones, and these are responsible for colour vision. The amount of light absorbed at each wavelength by each cone type defines its *tuning curve*, as shown in Figure 9.4. These cones are labelled L, M and S, supposedly named after long, medium and short wavelengths. However, this is a misleading nomenclature, because the L and M cones have very similar tuning curves, and both cone types respond best to light which looks greeny-yellow. In contrast, the S-cones are sensitive to short wavelength light which appears blue. Despite this misnaming, for simplicity, we will refer to them as red, green and blue cones.

The outputs of the photoreceptors are transmitted to the brain via one million fibres which comprise the *optic nerve*. However, trying to squeeze the outputs of 126 million photoreceptors into the optic nerve's one million fibres is the cause of a severe bottleneck.

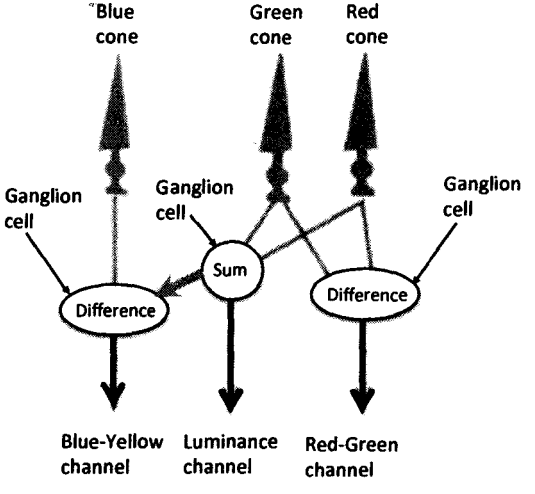


Figure 9.5. Combining outputs from three cone types to produce three new colour channels (ganglion cell outputs). From Frisby and Stone (2010)[16].

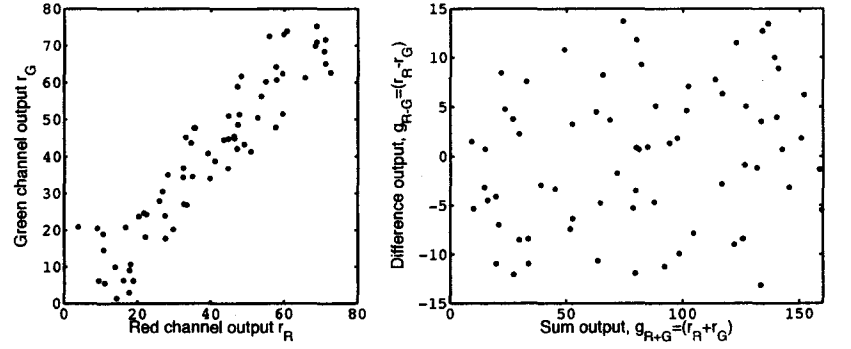


Figure 9.6. Schematic illustration of sum–difference recoding. (a) The similar tuning curves of red and green cones means that nearby cones have similar output values. (b) The recoded sum $g_{R+G} = r_R + r_G$ and difference $g_{R-G} = r_R - r_G$ channels are independent. From Stone, 2012[51].

Because the red and green cone types have similar tuning curves, neighbouring red and green cones usually have similar outputs. If these red and green cones had their own private nerve fibres to send their outputs along then these outputs would be similar; almost as if the same message is being sent along two different wires. Given the information bottleneck, this would obviously be a wasteful use of the capacity of the communication channels (i.e. nerve fibres).

One simple way to ensure that the messages in different optic nerve fibres are uncorrelated consists of using one fibre to carry the *sum* of the red and green cone outputs, and another to carry the *difference* between their outputs, as in Figure 9.5. For brevity, we will call this type of recoding *sum–difference recoding*; it is also obtained by applying *principal component analysis* to the data in Figure 9.6a. It can be shown that if cone outputs have Gaussian distributions then sum–difference recoding yields independent messages. Physical evidence for sum–difference recoding exists in the form of *ganglion cells*. These cells reside in the retina, where they collate the outputs of neighbouring photoreceptors, and each ganglion cell output is carried by one of the million nerve fibres in the optic nerve.

To confirm that the sum and difference signals are independent, we can test whether one can be *predicted* from the other. For example, if the two cone outputs are represented as r_R and r_G then the value of r_R tells us roughly what the value of r_G is; they are highly correlated, as shown in Figure 9.6a. Now, using g to denote ganglion cell output, suppose instead that these values are recoded in terms of a *sum* ganglion cell output, $g_{R+G} = r_R + r_G$, and a *difference* ganglion cell output, $g_{R-G} = r_R - r_G$, as shown in Figure 9.6b. The almost uniform coverage in Figure 9.6b suggests that the ganglion cell outputs g_{R+G} and g_{R-G} are independent, and this, in turn, suggests that ganglion cell channel capacity is not being wasted. In fact, ganglion cell channel capacity is only fully utilised if the outputs of each ganglion cell are also independent over time, and there is evidence for this[37]. In summary, using separate ganglion cells to transmit outputs of red and green cones necessarily wastes some of the channel capacity of ganglion cells, whereas using ganglion cells to implement sum–difference recoding of cone outputs using does not.

The Brain: An Efficient Encoder

The experiments and analyses described above suggest that the brain's ability to process information is about as efficient as it possibly can be. More importantly, information-theoretic analyses of such experiments have led to the general conclusion that, within sensory systems:

> ... information rates are very large, close to the physical limits imposed by the spike train entropy.
> Rieke *et al*, 1997[43].

Without information theory, we would have no way of telling how well neurons perform, because we would have little idea of what it means to measure neuronal information processing performance in absolute terms. And so we would not be able to tell that the answer to the question, "are brains good at processing information?" is *yes*. More importantly, we could not know that, within the constraints imposed by their physical structure, brains operate close to the limits defined by Shannon's mathematical theory of communication.

9.7. A Very Short History of Information Theory

Even the most gifted scientist cannot command an original theory out of thin air. Just as Einstein could not have devised his theories of relativity if he had no knowledge of Newton's work, so Shannon could not have created information theory if he had no knowledge of the work of Boltzmann (1875) and Gibbs (1902) on thermodynamic entropy, Wiener (1927) on signal processing, Nyquist (1928) on sampling theory, or Hartley (1928) on information transmission[40].

Even though Shannon was not alone in trying to solve one of the key scientific problems of his time (i.e. how to define and measure information), he was alone in being able to produce a complete mathematical theory of information: a theory that might otherwise have taken decades to construct. In effect, Shannon single-handedly accelerated the rate of scientific progress, and it is entirely possible that, without his contribution, we would still be treating information as if it were some ill-defined vital fluid.

9.8. Summary

In 1986, the physicist John Wheeler said:

> It is my opinion that everything must be based on a simple idea. And ... this idea, once we have finally discovered it, will be so compelling, so beautiful, that we will say to one another, yes, how could it have been any different?

So compelling, and so beautiful: information theory represents a fundamental insight that must surely rank as a candidate for Wheeler's "simple idea". Indeed, after many years of studying physics and information theory, Wheeler came up with a proposal which is both radical and intriguing:

> ... the universe is made of information; matter and energy are only incidental.

Insofar as it must be made of something, a universe in which all forms of energy and matter are simply different manifestations of pure information might be as sublime as this one.

Bibliography

[1] Applebaum, D. (2008). *Probability and Information: An Integrated Approach*. 2nd edition. Cambridge University Press.

[2] Avery, J. (2012). *Information Theory and Evolution*. World Scientific Publishing, New Jersey.

[3] Baeyer, H. (2005). *Information: The New Language of Science*. Harvard University Press, Cambridge, MA.

[4] Baldwin, J. (1896). A new factor in evolution. *The American Naturalist*, 30(354):441–451.

[5] Barlow, H. (1961). Possible principles underlying the transformation of sensory messages. In W.A. Rosenblith, editor, *Sensory Communication*, pp. 217–234. MIT Press, Cambridge, MA.

[6] Bennett, C. (1987). Demons, engines and the second law. *Scientific American*, 257(5):108–116.

[7] Bérut, A., Arakelyan, A., Petrosyan, A., Ciliberto, S., Dillenschneider, R., and Lutz, E. (2012). Experimental verification of Landauer's principle linking information and thermodynamics. *Nature*, 483(7388):187–189.

[8] Bialek, W. (2012). *Biophysics: Searching for Principles*. Princeton University Press, New Jersey.

[9] Bishop, C. (2006). *Pattern Recognition and Machine Learning*. Springer.

[10] Brown, P.F., Pietra, V.J.D., Mercer, R.L., Pietra, S.A.D. and Lai, J.C. (1992). An estimate of an upper bound for the entropy of English. *Computational Linguistics*, 1(18):31–40.

[11] Cover, T. and Thomas, J. (1991). *Elements of Information Theory*. John Wiley and Sons, New York.

[12] Darwin, C. (1859). *On the origin of species by means of natural selection, or the preservation of favoured races in the struggle for life*. 1st edition. John Murray, London.

[13] DeGroot, M. (1986). *Probability and Statistics*. 2nd edition. Addison-Wesley, New York.

[14] Deutsch, D. and Marletto, C. (2014). Reconstructing physics: The universe is information. *New Scientist*, 2970:30.

[15] Feynman, R., Leighton, R., and Sands, M. (1964). *Feynman Lectures on Physics*. Basic Books, New York.

[16] Frisby, JP and Stone, JV. (2010). *Seeing: The computational approach to biological vision*. MIT Press, Cambridge, MA.

[17] Friston, K. (2010). The free-energy principle: a unified brain theory? *Nature Review Neuroscience*, 11(2):127–138.

[18] Gabor, D. (1951). Lectures on communication theory. Technical report, Massachusetts Institute of Technology.

[19] Gatenby, R. and Frieden, B. (2013). The critical roles of information and nonequilibrium thermodynamics in evolution of living systems. *Bulletin of Mathematical Biology*, 75(4):589–601.

[20] Gibbs, JW. (1902). *Elementary Principles in Statistical Mechanics.* Charles Scribner's Sons, New York.

[21] Gleick, J. (2012). *The Information*. Vintage, London.

[22] Greene, B. (2004). *The Fabric of the Cosmos.* Knopf, New York.

[23] Grover, L. (1996). A fast quantum mechanical algorithm for database search. *Proceedings, 28th Annual ACM Symposium on the Theory of Computing*, pp. 212–219.

[24] Guizzo, E. (2003). The essential message: Claude Shannon and the making of information theory. MSc Thesis, Massachusetts Institute of Technology.
`http://dspace.mit.edu/bitstream/handle/1721.1/39429/54526133.pdf`

[25] Holland, J. (1992). *Adaptation in Natural and Artificial Systems*. MIT Press, Cambridge, MA.

[26] Jaynes, E. and Bretthorst, G. (2003). *Probability Theory: The Logic of Science*. Cambridge University Press, Cambridge, England.

[27] Jessop, A. (1995). *Informed Assessments: An Introduction to Information, Entropy and Statistics*. Ellis Horwood, London.

[28] Kolmogorov, A. (1933). *Foundations of the Theory of Probability*. English translation, 1956. Chelsea Publishing Company.

[29] Kostal, L., Lansky, P., and Rospars, J.-P. (2008). Efficient olfactory coding in the pheromone receptor neuron of a moth. *PLoS Computational Biology*, 4(4).

[30] Landauer, R. (1961). Irreversibility and heat generation in the computing process. *IBM J. Research and Development*, 5:183–191.

[31] Laughlin, S. (1981). A simple coding procedure enhances a neuron's information capacity. *Z Naturforsch C*, 36(9–10):910–912.

[32] Lemon, D. (2013). *A Student's Guide to Entropy*. Cambridge University Press, Cambridge, England.

[33] MacKay, A. (1967). Optimization of the genetic code. *Nature*, 216:159–160.

[34] MacKay, D. (2003). *Information Theory, Inference, and Learning Algorithms*. Cambridge University Press, Cambridge, England.

[35] Nemenman, I., Lewen, G., Bialek, W., and de Ruyter van Steveninck, R. (2008). Neural coding of natural stimuli: Information at sub-millisecond resolution. *PLoS Computational Biology*, 4(3).

[36] Nemenman, I., Shafee, F., and Bialek, W. (2002). Entropy and inference, revisited. In T.G. Dietterich, S. Becker, and Z. Ghahramani, editors, *Advances in Neural Information Processing Systems 14*, pp. 471–478. MIT Press, Cambridge, MA.

[37] Nirenberg, S and Carcieri, SM and Jacobs, AL and Latham, PE. (2001). Retinal ganglion cells act largely as independent encoders. *Nature*, 411(6838):698–701.

[38] Olshausen, B. and Field, D. (1996). Natural image statistics and efficient coding. *Network: Computation in Neural Systems*, 7:333–339.

[39] Paninski, L. (2003). Estimation of entropy and mutual information. *Neural Computation*, 15(6):1191–1253.

[40] Pierce, J. (1961). *An Introduction to Information Theory: Symbols, Signals and Noise*. 2nd edition, Dover, 1980.

[41] Reza, F. (1961). *Information Theory*. McGraw-Hill, New York.

[42] Rieke, F., Bodnar, D., and Bialek, W. (1995). Naturalistic stimuli increase the rate and efficiency of information transmission by primary auditory afferents. *Proceedings of the Royal Society of London. Series B: Biological Sciences*, 262(1365):259–265.

[43] Rieke, F., Warland, D., van Steveninck, R., and Bialek, W. (1997). *Spikes: Exploring the Neural Code*. MIT Press, Cambridge, MA.

[44] Schneider, T. D. (2010). 70% efficiency of bistate molecular machines explained by information theory, high dimensional geometry and evolutionary convergence. *Nucleic acids research*, 38(18):5995–6006.

[45] Schürmann, T. and Grassberger, P. (1996). Entropy estimation of symbol sequences. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 6(3):414–427.

[46] Seife, C. (2007). *Decoding the Universe: How the New Science of Information Is Explaining Everything in the Cosmos, From Our Brains to Black Holes*. Penguin.

[47] Sengupta, B., Stemmler, M., and Friston, K. (2013). Information and efficiency in the nervous system: a synthesis. *PLoS Computational Biology*, 9(7).

[48] Shannon, C. (1948). A mathematical theory of communication. *Bell System Technical Journal*, 27:379–423.

[49] Shannon, C. (1951). Prediction and entropy of printed English. *Bell System Technical Journal*, 30:47–51.

[50] Shannon, C. and Weaver, W. (1949). *The Mathematical Theory of Communication*. University of Illinois Press.

[51] Stone, J. (2012). *Vision and Brain: How we perceive the world*. MIT Press, Cambridge, MA.

[52] Stone, J. (2013). *Bayes' Rule: A Tutorial Introduction to Bayesian Analysis*. Sebtel Press, Sheffield, England.

[53] Wallis, K. (2006). A note on the calculation of entropy from histograms. Technical report, University of Warwick.

[54] Watkins, C. (2008). Selective breeding analysed as a communication channel: channel capacity as a fundamental limit on adaptive complexity. In *Symbolic and Numeric Algorithms for Scientific Computing: Proceedings of SYNASC'08*, pp. 514–518.

[55] Zhaoping, L. (2014). *Understanding Vision: Theory, Models, and Data*. Oxford University Press.

Index

About the Author.
Dr James Stone is a Reader in Computational Neuroscience at the University of Sheffield, England. Previous books are listed below.

Bayes' Rule: A Tutorial Introduction to Bayesian Analysis,
JV Stone, Sebtel Press, 2013.
Vision and Brain: How We Perceive the World,
JV Stone, MIT Press, 2012.
Seeing: The Computational Approach to Biological Vision,
JP Frisby and JV Stone, MIT Press, 2010.
Independent Component Analysis: A Tutorial Introduction,
JV Stone, MIT Press, 2004.