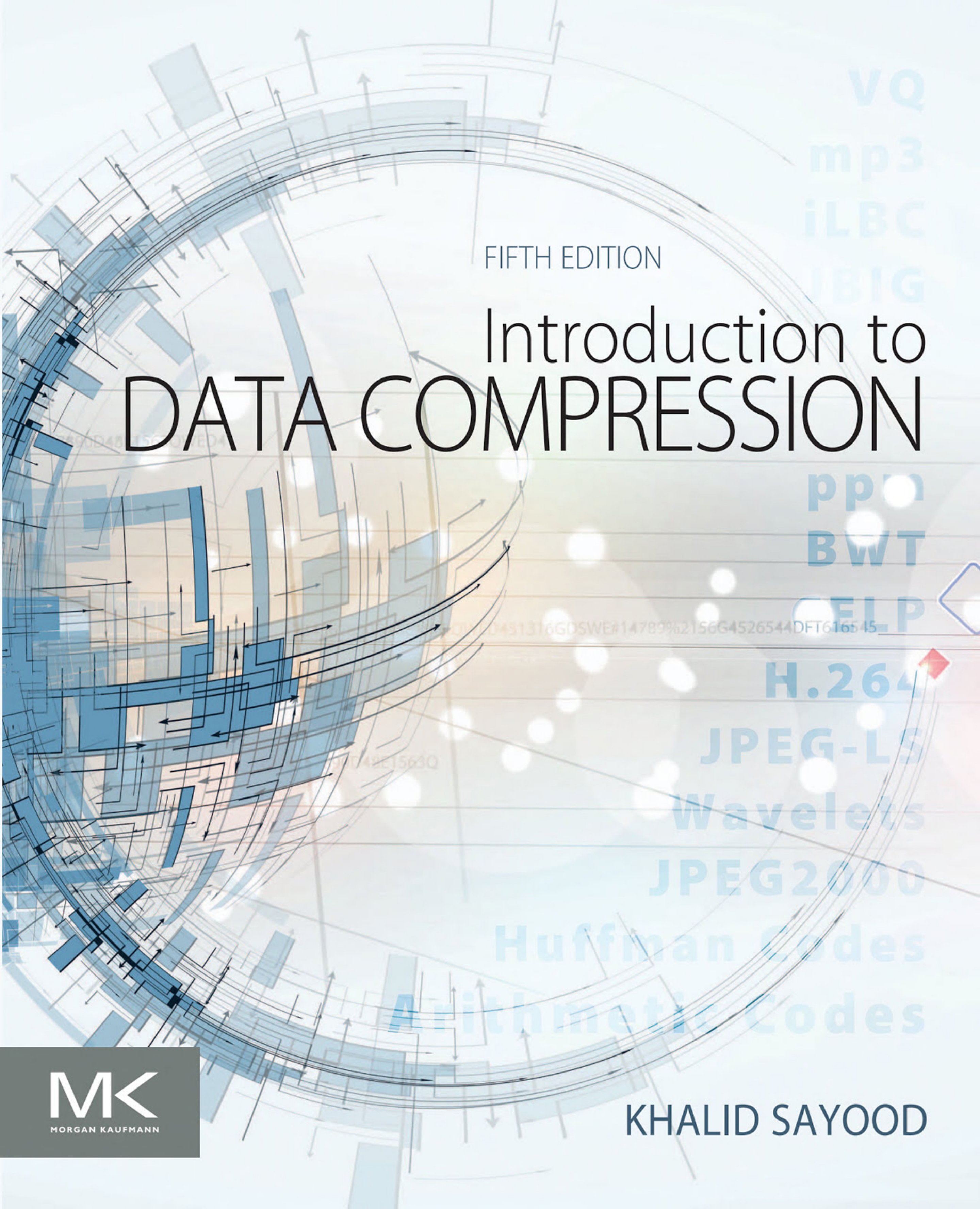
FIFTH EDITION

Introduction to DATA COMPRESSION



KHALID SAYOOD

MORGAN KAUFMANN

Introduction to Data Compression

Fifth Edition

Introduction to Data Compression

Fifth Edition

Khalid Sayood

ELSEVIER

MORGAN KAUFMANN PUBLISHERS

AN IMPRINT OF ELSEVIER

Morgan Kaufmann is an imprint of Elsevier
50 Hampshire Street, 5th Floor, Cambridge, MA 02139, United States

Library of Congress Cataloging-in-Publication Data
A catalog record for this book is available from the Library of Congress

British Library Cataloguing-in-Publication Data
A catalogue record for this book is available from the British Library

ISBN: 978-0-12-809474-7

Publisher: Katey Birtcher
Acquisition Editor: Steve Merken
Editorial Project Manager: Peter Jardim
Production Project Manager: Punithavathy Govindaradjane
Designer: Matthew Limbert

Typeset by VTeX

To Füsun

Preface

Data compression has been an enabling technology for the information revolution. As this revolution has changed our lives, data compression has become a more and more ubiquitous, if often invisible, presence. From smartphones, to digital television and movies, data compression is an integral part of information technology and of our lives. This incorporation of compression into more and more of our lives also points to a certain degree of maturation and stability of the technology. This maturity is reflected in the fact that there are fewer differences between each edition of this book. In the second edition we had added new techniques that had been developed since the first edition of this book came out. In the third edition we added a chapter on audio compression, a topic that had not been adequately covered in the second edition. The fourth edition expanded the coverage of wavelets. In this edition we have added details of standards that have come out since the last edition such as High Efficiency Video Coding and new standards from the JPEG committee. We have added a description of Grammar based codes which might become more important in the coming years and as always we have filled in details left out from the previous editions. I have also indulged my latest interests by including a few applications of data compression algorithms to bioinformatics.

Though the percentage growth of this edition is less than that of previous editions it is still growth and it has yet again enlarged the book. Despite its expanded coverage, the intent remains the same: to provide an introduction to the art or science of data compression. There is a tutorial description of most of the popular compression techniques followed by a description of how these techniques are used for image, speech, text, audio, and video compression. We have attempted to make the chapters on techniques as self contained as possible. This is of course not completely possible for the application chapters which rely on multiple techniques presented in previous chapters, but here too we have tried to make sure that the dependence does not interrupt the explanation. One hopes the size of the book will not be intimidating and once you open the book and begin reading a particular section we hope you will find the content easily accessible. If some material is not clear please write to me at ksayood@unl.edu with specific questions and I will try and help (homework problems and projects are of course your responsibility).

AUDIENCE

If you are designing hardware or software implementations of compression algorithms, or need to interact with individuals engaged in such design, or are involved in development of multimedia applications and have some background in either electrical or computer engineering, or computer science, this book should be useful to you. We have included a large number of examples to aid in self-study. We have also included discussion of various multimedia standards. The intent here is not to provide all the details that may be required to implement a standard but to provide information that will help you follow and understand the standards documents. The final authority is always the standards document.

COURSE USE

The impetus for writing this book came from the need for a self-contained book that could be used at the senior/graduate level for a course in data compression in either electrical engineering, computer engineering, or computer science departments. There are problems and project ideas after most of the chapters. A solutions manual is available from the publisher. Also at http://datacompression.unl.edu we provide links to various course homepages, which can be a valuable source of project ideas and support material.

The material in this book is too much for a one semester course. However, with judicious use of the starred sections, this book can be tailored to fit a number of compression courses that emphasize various aspects of compression. If the course emphasis is on lossless compression, the instructor could cover most of the sections in the first seven chapters. Then, to give a taste of lossy compression, the instructor could cover Sections 9.1–9.5 of Chapter 9, followed by Chapter 13 and its description of JPEG, and Chapter 19, which describes video compression approaches used in multimedia communications. If the class interest is more attuned to audio compression, then instead of Chapters 13 and 19, the instructor could cover Chapters 14 and 17. If the latter option is taken, depending on the background of the students in the class, Chapter 12 may be assigned as background reading. If the emphasis is to be on lossy compression, the instructor could cover Chapter 2, the first two sections of Chapter 3, Sections 4.4 and 4.6 of Chapter 4 (with a cursory overview of Sections 4.2 and 4.3), Chapter 8, selected parts of Chapter 9, and Chapter 10 through 16. At this point depending on the time available and the interests of the instructor and the students portions of the remaining three chapters can be covered. I have always found it useful to assign a term project in which the students can follow their own interests as a means of covering material that is not covered in class but is of interest to the student.

APPROACH

In this book, we cover both lossless and lossy compression techniques with applications to image, speech, text, audio, and video compression. The various lossless and lossy coding techniques are introduced with just enough theory to tie things together. The necessary theory is introduced just before we need it. Therefore, there are three *mathematical preliminaries* chapters. In each of these chapters, we present the mathematical material needed to understand and appreciate the techniques that follow.

Although this book is an introductory text, the word *introduction* may have a different meaning for different audiences. We have tried to accommodate the needs of different audiences by taking a dual-track approach. Wherever we felt there was material that could enhance the understanding of the subject being discussed but could still be skipped without seriously hindering your understanding of the technique, we marked those sections with a star (★). If you are primarily interested in understanding how the various techniques function, especially if you are using this book for self-study, we recommend you skip the starred sections, at least in a first reading. Readers who require a slightly more theoretical approach should use the starred sections. Except for the starred sections, we have tried to keep the mathematics to a minimum.

LEARNING FROM THIS BOOK

I have found that it is easier for me to understand things if I can see examples. Therefore, I have relied heavily on examples to explain concepts. You may find it useful to spend more time with the examples if you have difficulty with some of the concepts.

Compression is still largely an art and to gain proficiency in an art we need to get a "feel" for the process. We have included software implementations for most of the techniques discussed in this book, along with a large number of data sets. The software and data sets can be obtained from ftp://ftp.mkp.com/pub/Sayood/ or from datacompression.unl.edu. The programs are written in C and have been tested on a number of platforms. The programs should run under most flavors of UNIX machines and, with some slight modifications, under other operating systems as well. More detailed information is contained in the README file in the *pub/Sayood* directory.

You are strongly encouraged to use and modify these programs to work with your favorite data in order to understand some of the issues involved in compression. A useful and achievable goal should be the development of your own compression package by the time you have worked through this book. This would also be a good way to learn the trade-offs involved in different approaches. We have tried to give comparisons of techniques wherever possible; however, different types of data have their own idiosyncrasies. The best way to know which scheme to use in any given situation is to try them.

CONTENT AND ORGANIZATION

The organization of the chapters is as follows: We introduce the mathematical preliminaries necessary for understanding lossless compression in Chapter 2; Chapters 3 and 4 are devoted to coding algorithms, including Huffman coding, arithmetic coding, Golomb–Rice codes, and Tunstall codes. Chapters 5 and 6 describe many of the popular lossless compression schemes along with their applications. The schemes include LZW, *ppm*, BWT, and DMC, among others. In Chapter 7 we describe a number of lossless image compression algorithms and their applications in a number of international standards. The standards include the JBIG standards and various facsimile standards.

Chapter 8 is devoted to providing the mathematical preliminaries for lossy compression. Quantization is at the heart of most lossy compression schemes. Chapters 9 and 10 are devoted to the study of quantization. Chapter 9 deals with scalar quantization, and Chapter 10 deals with vector quantization. Chapter 11 deals with differential encoding techniques, in particular differential pulse code modulation (DPCM) and delta modulation. Included in this chapter is a discussion of the CCITT G.726 standard.

Chapter 12 is our third mathematical preliminaries chapter. The goal of this chapter is to provide the mathematical foundation necessary to understand some aspects of the transform, subband, and wavelet-based techniques that are described in the next four chapters. As in the case of the previous mathematical preliminaries chapters, not all material covered is necessary for everyone. We describe the JPEG standard in Chapter 13, the CCITT G.722 international standard in Chapter 14, and EZW, SPIHT, and JPEG 2000 in Chapter 16.

Chapter 17 is devoted to audio compression. We describe the various MPEG audio compression schemes in this chapter including the scheme popularly known as *mp3*.

Chapter 18 covers techniques in which the data to be compressed are analyzed, and a model for the generation of the data is transmitted to the receiver. The receiver uses this model to synthesize the data. These analysis/synthesis and analysis by synthesis schemes include linear predictive schemes used for low-rate speech coding and the fractal compression technique. We describe the federal government LPC-10 standard. Code-excited linear prediction (CELP) is a popular example of an analysis by synthesis scheme. We also discuss three CELP-based standards, the federal standard 1016, the G.728 international standard, and the wideband speech compression standard G.722.2 as well as a the 2.4 kbps mixed excitation linear prediction (MELP) technique. We have also included an introduction to three speech compression standards currently in use for speech compression for Internet applications: the Internet Low Bandwidth Coder, SILK, and ITU-T G.729 standard.

Chapter 19 deals with video coding. We describe popular video coding techniques via description of various international standards, including the High Efficiency Video Coding (HEVC) or H.265 standard.

A PERSONAL VIEW

For me, data compression is more than a manipulation of numbers; it is the process of discovering the information that is contained in the data. Sometimes the information is clear and evident, sometimes it is hidden or occult. (My lab is called the Occult Information Lab (OIL) which from time to time has lead to some misunderstandings.) The process of discovering this information is a source of joy. Encoding this information requires modeling the structures in the data. To see this in a very different context consider the following brief poems from different centuries, different cultures, written in different languages.

Indeed, indeed, repentance oft before
I swore but was I sober when I swore?
And then came Spring and rose in hand
my threadbare penitence a-pieces tore.

Omar Khayyam, Persia, 1048–1131. Translated by Edward Fitzgerald

Even in Kyoto
hearing the cuckoo's cry
I long for Kyoto.

Matsuo Basho, Japan, 1644–1694. Translated by Robert Haas

To explain these few lines would take volumes. Despite being from different centuries and cultures, they tap into a common human experience so that in our mind's eye, we can reconstruct what the poet was trying to say. To understand the words we need a model of reality that is close to that of the poet. The genius of the poet lies in identifying a model of reality that is so much a part of our humanity that centuries later and in widely diverse cultures, these few words can evoke volumes.

Data compression is much more limited in its aspirations, and it may be presumptuous to mention it in the same breath as poetry. But there is much that is similar to both endeavors. Data compression involves identifying models for the many different types of structures that exist in different types of data and then using these models, perhaps along with the perceptual framework in which these data will be used, to obtain a compact representation of the data. These structures can be in the form of patterns that we can recognize simply by plotting the data, or they might be structures that require a more abstract approach to comprehend. Often, it is not the data but the structure within the data that contains the information, and the development of data compression involves the discovery of these structures.

In *The Long Dark Teatime of the Soul* by Douglas Adams, the protagonist finds that he can enter Valhalla (a rather shoddy one) if he tilts his head in a certain way. Appreciating the structures that exist in data sometimes require us to tilt our heads in a certain way. There are an infinite number of ways we can tilt our head and, in order not to get a pain in the neck (carrying our analogy to absurd limits), it would be nice to know some of the ways that will generally lead to a profitable result. One of the objectives of this book is to provide you with a frame of reference that can be used for further exploration. I hope this exploration will provide as much enjoyment for you as it has given to me.

ACKNOWLEDGMENTS

It has been a lot of fun writing and revising this book. My task has been made considerably easier and the end product considerably better because of the help I have received. Acknowledging that help is itself a pleasure.

The first edition benefitted from the careful and detailed criticism of Roy Hoffman from IBM, Glen Langdon from the University of California at Santa Cruz, Debra Lelewer from California Polytechnic State University, Eve Riskin from the University of Washington, Ibrahim Sezan from Kodak, and Peter Swaszek from the University of Rhode Island. They provided detailed comments on all or most of the first edition. Nasir Memon then at Polytechnic University, Victor Ramamoorthy then at S3, Grant Davidson at Dolby Corporation, Hakan Caglar, who was then at TÜBITAK in Istanbul, and Allen Gersho from the University of California at Santa Barbara reviewed parts of the manuscript.

For the second edition Steve Tate at the University of North Texas, Sheila Horan at New Mexico State University, Edouard Lamboray at Oerlikon Contraves Group, Steven Pigeon at the University of Montreal, and Jesse Olvera at Raytheon Systems reviewed the entire manuscript. Emin Anarım of Boğaziçi University and Hakan Çağlar helped me with the development of the chapter on wavelets. Mark Fowler provided extensive comments on Chapters 12–15, correcting mistakes of both commission and omission. Tim James, Devajani Khataniar, and Lance Pérez also read and critiqued parts of the new material in the second edition. Chloeann Nelson, along with trying to stop me from splitting infinitives, also tried to make the first two editions of the book more user-friendly. The third edition benefitted from the critique of Rob Maher, now at Montana State, who generously gave of his time to help with the chapter on audio compression. The fourth edition was much improved by the comments of Mark Fowler from SUNY Binghamton and Pierre Jouvelet from the Ecole Superieure des Mines de Paris. The aforementioned wavelet material was added because Mark thought that it should be there. Pierre's detailed and extensive review was extremely helpful. The chapter on Wavelet Image Compression benefitted from the review of Mike Marcellin of the University of Arizona. Mike agreed to look at the chapter while in the midst of end-of-semester crunch, which is an act of friendship those in the teaching profession will appreciate. Mike is a gem. Mark also reviewed the material in the fifth edition. This edition was also reviewed by Kamisetty R. Rao of the University of Texas at Arlington, and Jason McNeely of the University of Alaska. I have tried to follow Jason's suggestion to improve the outline sections in the various chapter.

Since the appearance of the first edition, various readers have sent me their comments and critiques. I am grateful to all who sent me comments and suggestions. I am especially grateful to Roberto Lopez-Hernandez, Dirk vom Stein, Christopher A. Larrieu, Ren Yih Wu, Humberto D'Ochoa, Roderick Mills, Mark Elston, Jeerasuda Keesorth, and Paul Palmer for pointing out errors and suggesting improvements to the book. I am also grateful to the various instructors who have sent me their critiques. In particular

I would like to thank Bruce Bomar from the University of Tennessee, Ralph Wilkerson from the University of Missouri–Rolla, Adam Drozdek from Duquesne University, Ed Hong and Richard Ladner from the University of Washington, Lars Nyland from the Colorado School of Mines, Mario Kovac from the University of Zagreb, Jim Diamond of Acadia University, Haim Perlmutter from Ben-Gurion University, Dan Tamir from Texas State University, Luis Cruz from the University of Coimbra, and John Woods from Rensselaer Polytechnic Institute. Paul Amer, from the University of Delaware, has been one of my earliest, most consistent, and most welcome critics. His courtesy is greatly appreciated.

Frazer Williams and Mike Hoffman, from my department at the University of Nebraska, provided reviews for the first edition of the book. Mike has continued to provide me with guidance and has read and critiqued the new chapters in every edition of the book. I rely heavily on his insights and his critique and would be lost without him. It is nice to have friends of his intellectual caliber and generosity.

Pat Worster edited many of the chapters and tried to teach me the proper use of the semi-colon, and to be a bit more generous with commas. The book reads a lot better because of her attention. With all this help one would expect a perfect book. The fact that it is not is a reflection of my imperfection.

Stephen Merken and Peter Jardim from Elsevier have been responsible for this revision. Peter has had the thankless task of dealing with my never-ending excuses. Punithavathy Govindaradjane was responsible for trying to bring the project to a timely conclusion.

Over the years a number of my students have had a direct impact on this book. Most of the examples in this book were generated in a lab set up by Andy Hadenfeldt. James Nau helped me extricate myself out of numerous software puddles giving freely of his time. In my times of panic, he has always been just an email or voice mail away. Zhao Li wrote the solutions to the problems in the second edition. Sam Way helped me out with several of the examples in the fourth edition and Dave Russell provided me with helpful criticism after trying to teach out of this book. Ufuk Nalbantoğlu continues to teach me new ways of looking at data compression.

I would like to thank the various "models" for the data sets that accompany this book and were used as examples. The individuals in the images are Sinan Sayood, Sena Sayood, and Elif Sevuktekin. The female voice belongs to Pat Masek.

This book reflects what I have learned over the years. I have been very fortunate in the teachers I have had. David Farden, now at North Dakota State University, introduced me to the area of digital communication. Norm Griswold formerly at Texas A&M University introduced me to the area of data compression. Jerry Gibson, now at the University of California at Santa Barbara was my Ph.D. advisor and helped me get started on my professional career. He continues in his attempts to educate me and to stop me from embarrassing myself. I am grateful for his gracious persistence. I have also learned a lot from my students at the University of Nebraska and Boğaziçi University. Their interest and curiosity forced me to learn and kept me in touch with the broad field that is data compression today. I learned at least as much from them as they learned from me.

Much of this learning would not have been possible but for the support I received from NASA. The late Warner Miller and Pen-Shu Yeh at the Goddard Space Flight Center and Wayne Whyte at what was then the Lewis Research Center were a source of support and ideas. I am truly grateful for their helpful guidance, trust, and friendship.

Our two boys, Sena and Sinan, graciously forgave my evenings and weekends at work. They were tiny (witness the images) when I first started writing this book. They are young men now, as gorgeous to my eyes now as they have always been, and "the book" has been their (sometimes unwanted) companion through all these years. For their graciousness and for the great pleasure they have given me and continue to give me, I thank them.

Above all the person most responsible for the existence of this book is my partner and closest friend Füsun. Her support and her friendship gives me the freedom to do things I would not otherwise even consider. She centers my universe, is the color of my existence, and, as with every significant endeavor that I have undertaken since I met her, this book is at least as much hers as it is mine.

CHAPTER

INTRODUCTION

1

Peta, exa, zetta, yotta,[1] the number of bytes of data, stored, processed, and transmitted keeps soaring, and in the process, keeps transforming our world. This transformation includes the ever-present, ever-growing Internet; the explosive development of mobile communications; and the ever-increasing importance of video communication. Data compression is one of the enabling technologies for each of these aspects of the multimedia revolution. It would not be practical to put images, let alone audio and video, on websites if it were not for data compression algorithms. Cellular phones would not be able to provide communication with increasing clarity were it not for compression. The advent of digital TV would not be possible without compression. Data compression, which for a long time was the domain of a relatively small group of engineers and scientists, is now ubiquitous. Make a call on your cell phone, and you are using compression. Surf on the Internet, and you are using (or wasting) your time with assistance from compression. Listen to music or watch a movie, and you are being entertained courtesy of compression.

Data compression is the art or science of representing information in a compact form. We create these compact representations by identifying and using structures that exist in the data. Data can be characters in a text file, numbers that are samples of speech or image waveforms, or sequences of numbers that are generated by other processes. The reason we need data compression is that more and more of the information that we generate and use is in digital form—consisting of numbers represented by bytes of data. And the number of bytes required to represent multimedia data can be huge. For example, in order to digitally represent 1 second of video without compression (using the CCIR 601 format described in Chapter 19), we need more than 20 megabytes, or 160 megabits. If we consider the number of seconds in a movie, we can easily see why we would need compression. To represent 2 minutes of uncompressed CD-quality music (44,100 samples per second, 16 bits per sample) requires more than 84 million bits. Downloading music from a website at these rates would take a long time.

As human activity has a greater and greater impact on our environment, there is an ever-increasing need for more information about our environment, how it functions, and what we are doing to it. Various space agencies from around the world, including the European Space Agency (ESA), the National Aeronautics and Space Administration (NASA), the Canadian Space Agency (CSA), and the Japan Aerospace Exploration Agency (JAXA), are collaborating on a program to monitor global change that will generate half a terabyte of data per *day* when they are fully operational. New sequencing technologies are resulting in ever increasing size of databases containing genomic information while new medical scanning applications can result in the generation of petabytes of data.

Given the explosive growth of data that needs to be transmitted and stored, why not focus on developing better transmission and storage technologies? This is happening, but it is not enough. There have been significant advances that permit larger and larger volumes of information to be stored and

[1]Mega: 10^6, giga: 10^9, tera: 10^{12}, peta: 10^{15}, exa: 10^{18}, zetta: 10^{21}, yotta: 10^{24}.

Introduction to Data Compression. DOI: 10.1016/B978-0-12-809474-7.00001-X

transmitted without using compression. However, while it is true that both storage and transmission capacities are steadily increasing with new technological innovations, as a corollary to Parkinson's First Law,[2] it seems that the need for mass storage and transmission increases at least twice as fast as storage and transmission capacities improve. Then there are situations in which capacity has not increased significantly. For example, the amount of information we can transmit over the airwaves will always be limited by the characteristics of the atmosphere.

An early example of data compression is Morse code, developed by Samuel Morse in the mid-19th century. Letters sent by telegraph are encoded with dots and dashes. Morse noticed that certain letters occurred more often than others. In order to reduce the average time required to send a message, he assigned shorter sequences to letters that occur more frequently, such as e ($\cdot$) and a ($\cdot -$), and longer sequences to letters that occur less frequently, such as q ($- - \cdot -$) and j ($\cdot - - -$). This idea of using shorter codes for more frequently occurring characters is used in Huffman coding, which we will describe in Chapter 3.

Where Morse code uses the frequency of occurrence of single characters, a widely used form of Braille code, which was also developed in the mid-19th century, uses the frequency of occurrence of words to provide compression [1]. In Braille coding, 2×3 arrays of dots are used to represent text. Different letters can be represented depending on whether the dots are raised or flat. In Grade 1 Braille, each array of six dots represents a single character. However, given six dots with two positions for each dot, we can obtain 2^6, or 64, different combinations. If we use 26 of these for the different letters, we have 38 combinations left. In Grade 2 Braille, some of these leftover combinations are used to represent words that occur frequently, such as "and" and "for." One of the combinations is used as a special symbol indicating that the symbol that follows is a word and not a character, thus allowing a large number of words to be represented by two arrays of dots. These modifications, along with contractions of some of the words, result in an average reduction in space, or compression, of about 20% [1].

Statistical structure is being used to provide compression in these examples, but that is not the only kind of structure that exists in the data. There are many other kinds of structures existing in data of different types that can be exploited for compression. Consider speech. When we speak, the physical construction of our voice box dictates the kinds of sounds that we can produce. The mechanics of speech production impose a structure on speech. Therefore, instead of transmitting the speech itself, we could send information about the conformation of the voice box, which could be used by the receiver to synthesize the speech. An adequate amount of information about the conformation of the voice box can be represented much more compactly than the numbers that are the sampled values of speech. Therefore, we get compression. This compression approach is being used currently in a number of applications, including transmission of speech over cell phones and the synthetic voice in toys that speak. An early version of this compression approach, called the *vocoder* (*vo*ice *coder*), was developed by Homer Dudley at Bell Laboratories in 1936. The vocoder was demonstrated at the New York World's Fair in 1939, where it was a major attraction. We will revisit the vocoder and this approach to compression of speech in Chapter 18.

These are only a few of the many different types of structures that can be used to obtain compression. The structure in the data is not the only thing that can be exploited to obtain compression. We can

[2]Parkinson's First Law: "Work expands so as to fill the time available," in *Parkinson's Law and Other Studies in Administration*, by Cyril Northcote Parkinson, Ballantine Books, New York, 1957.

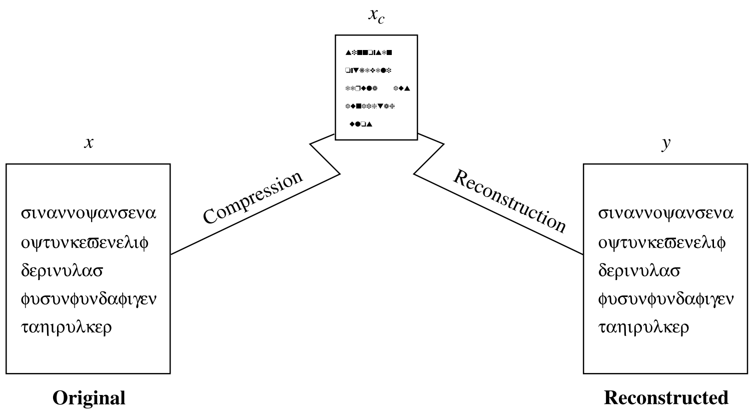


FIGURE 1.1

Compression and reconstruction.

also make use of the characteristics of the user of the data. Many times, for example, when transmitting or storing speech and images, the data are intended to be perceived by a human, and humans have limited perceptual abilities. For example, we cannot hear the very high frequency sounds that dogs can hear. If something is represented in the data that cannot be perceived by the user, is there any point in preserving that information? The answer is often "no." Therefore, we can make use of the perceptual limitations of humans to obtain compression by discarding irrelevant information. This approach is used in a number of compression schemes that we will visit in Chapters 13, 14, and 17.

Before we embark on our study of data compression techniques, let's take a general look at the area and define some of the key terms and concepts we will be using in the rest of the book.

1.1 COMPRESSION TECHNIQUES

When we speak of a compression technique or compression algorithm,[3] we are actually referring to two algorithms. There is the compression algorithm that takes an input $\mathcal{X}$ and generates a representation $\mathcal{X}_c$ that requires fewer bits, and there is a reconstruction algorithm that operates on the compressed representation $\mathcal{X}_c$ to generate the reconstruction $\mathcal{Y}$. These operations are shown schematically in Fig. 1.1.

[3]The word *algorithm* comes from the name of an early 9th-century Persian mathematician, Muhammad Ibn Musa Al-Khwarizmi. He was a scholar in the House of Wisdom in Baghdad during the Abbasid Caliphate where he wrote a treatise entitled *The Compendious Book on Calculation by* al-jabr *and* al-muqabala, in which he explored (among other things) the solution of various linear and quadratic equations via rules or an "algorithm." This approach became known as the method of Al-Khwarizmi. The name was changed to *algoritni* in Latin, from which we get the word *algorithm*. The name of the treatise also gave us the word *algebra* [2]. And his Book of Numbers introduced the positional numbering system to Europe three hundred years after his death.

We will follow convention and refer to both the compression and reconstruction algorithms together to mean the compression algorithm.

Based on the requirements of reconstruction, data compression schemes can be divided into two broad classes: *lossless* compression schemes, in which $\mathcal{Y}$ is identical to $\mathcal{X}$, and *lossy* compression schemes, which generally provide much higher compression than lossless compression but allow $\mathcal{Y}$ to be different from $\mathcal{X}$.

1.1.1 LOSSLESS COMPRESSION

Lossless compression techniques, as their name implies, involve no loss of information. If data have been losslessly compressed, the original data can be recovered exactly from the compressed data. Lossless compression is generally used for applications that cannot tolerate any difference between the original and reconstructed data.

Text compression is an important area for lossless compression. It is very important that the reconstruction is identical to the original text, as very small differences can result in statements with very different meanings. Consider the sentences "Do *not* send money" and "Do *now* send money." A similar argument holds for computer files and for certain types of data such as bank records.

If data of any kind are to be processed or "enhanced" later to yield more information, it is important that the integrity be preserved. For example, suppose we compressed a radiological image in a lossy fashion; and the difference between the reconstruction $\mathcal{Y}$ and the original $\mathcal{X}$ was visually undetectable. If this image was later enhanced, the previously undetectable differences may cause the appearance of artifacts that could seriously mislead the radiologist. Because the price for this kind of mishap may be a human life, it makes sense to be very careful about using a compression scheme that generates a reconstruction that is different from the original.

Data obtained from satellites often are processed later to obtain different numerical indicators of vegetation, deforestation, and so on. If the reconstructed data are not identical to the original data, processing may result in "enhancement" of the differences. It may not be possible to go back and obtain the same data over again. Therefore, it is not advisable to allow for any differences to appear in the compression process.

There are many situations that require compression where we want the reconstruction to be identical to the original. There are also a number of situations in which it is possible to relax this requirement in order to get more compression. In these situations, we look to lossy compression techniques.

1.1.2 LOSSY COMPRESSION

Lossy compression techniques involve some loss of information, and data that have been compressed using lossy techniques generally cannot be recovered or reconstructed exactly. In return for accepting this distortion in the reconstruction, we can generally obtain much higher compression ratios than is possible with lossless compression.

In many applications, this lack of exact reconstruction is not a problem. For example, when storing or transmitting speech, the exact value of each sample of speech is not necessary. Depending on the quality required of the reconstructed speech, varying amounts of loss of information about the value of each sample can be tolerated. If the quality of the reconstructed speech is to be similar to that heard on the telephone, a significant loss of information can be tolerated. However, if the reconstructed speech needs to be of the quality heard on a compact disc, the amount of information loss that can be tolerated is much lower.

Similarly, when viewing a reconstruction of a video sequence, the fact that the reconstruction is different from the original is generally not important as long as the differences do not result in annoying artifacts. Thus, video is generally compressed using lossy compression.

Once we have developed a data compression scheme, we need to be able to measure its performance. Because of the number of different areas of application, different terms have been developed to describe and measure the performance.

1.1.3 MEASURES OF PERFORMANCE

A compression algorithm can be evaluated in a number of different ways. We could measure the relative complexity of the algorithm, the memory required to implement the algorithm, how fast the algorithm performs on a given machine, the amount of compression, and how closely the reconstruction resembles the original. In this book we will mainly be concerned with the last two criteria. Let us take each one in turn.

A very logical way of measuring how well a compression algorithm compresses a given set of data is to look at the ratio of the number of bits required to represent the data before compression to the number of bits required to represent the data after compression. This ratio is called the *compression ratio*. Suppose storing an image made up of a square array of 256×256 pixels requires 65,536 bytes. The image is compressed and the compressed version requires 16,384 bytes. We would say that the compression ratio is 4:1. We can also represent the compression ratio by expressing the reduction in the amount of data required as a percentage of the size of the original data. In this particular example, the compression ratio calculated in this manner would be 75%.

Another way of reporting compression performance is to provide the average number of bits required to represent a single sample. This is generally referred to as the *rate*. For example, in the case of the compressed image described above, the average number of bits per pixel in the compressed representation is 2. Thus, we would say that the rate is 2 bits per pixel.

In lossy compression, the reconstruction differs from the original data. Therefore, in order to determine the efficiency of a compression algorithm, we have to have some way of quantifying the difference. The difference between the original and the reconstruction is often called the *distortion*. (We will describe several measures of distortion in Chapter 8.) Lossy techniques are generally used for the compression of data that originate as analog signals, such as speech and video. In compression of speech and video, the final arbiter of quality is human. Because human responses are difficult to model mathematically, many approximate measures of distortion are used to determine the quality of the reconstructed waveforms. We will discuss this topic in more detail in Chapter 8.

Other terms that are also used when talking about differences between the reconstruction and the original are *fidelity* and *quality*. When we say that the fidelity or quality of a reconstruction is high, we mean that the difference between the reconstruction and the original is small. Whether this difference is a mathematical difference or a perceptual difference should be evident from the context.

1.2 MODELING AND CODING

While reconstruction requirements may force the decision of whether a compression scheme is to be lossy or lossless, the exact compression scheme we use will depend on a number of different factors. Some of the most important factors are the characteristics of the data that need to be compressed. A compression technique that will work well for the compression of text may not work well for compressing images. Each application presents a different set of challenges.

There is a saying attributed to a whole slew of people from Mark Twain to Warren Buffet, “If the only tool you have is a hammer, you approach every problem as if it were a nail.” Our intention in this book is to provide you with a large number of tools that you can use to solve a particular data compression problem. It should be remembered that data compression, if it is a science at all, is an experimental science. The approach that works best for a particular application will depend to a large extent on the redundancies inherent in the data.

The development of data compression algorithms for a variety of data can be divided into two phases. The first phase is usually referred to as *modeling*. In this phase, we try to extract information about any redundancy that exists in the data and describe the redundancy in the form of a model. The second phase is called *coding*. A description of the model and a “description” of how the data differ from the model are encoded, generally using a binary alphabet. The difference between the data and the model is often referred to as the *residual*. In the following three examples, we will look at three different ways that data can be modeled. We will then use the model to obtain compression.

Example 1.2.1. Consider the following sequence of numbers $\{x_1, x_2, x_3, \dots\}$:

9	11	11	11	14	13	15	17	16	17	20	21

If we were to transmit or store the binary representations of these numbers, we would need to use 5 bits per sample. However, by exploiting the structure in the data, we can represent the sequence using fewer bits. If we plot these data as shown in Fig. 1.2, we see that the data seem to fall on a straight line. A model for the data could, therefore, be a straight line given by the equation

$$\hat{x}_n = n + 8 \quad n = 1, 2, \dots$$

The structure in this particular sequence of numbers can be characterized by an equation. Thus, $\hat{x}_1 = 9$, while $x_1 = 9$, $\hat{x}_2 = 10$, while $x_2 = 11$, and so on. To make use of this structure, let's examine the difference between the data and the model. The difference (or residual) is given by the sequence

$$e_n = x_n - \hat{x}_n \; : \quad 0\,1\,0\; -1\,1\; -1\,0\,1\; -1\; -1\,1\,1$$

The residual sequence consists of only three numbers $\{-1, 0, 1\}$. If we assign a code of 00 to -1, a code of 01 to 0, and a code of 10 to 1, we need to use 2 bits to represent each element of the residual sequence. Therefore, we can obtain compression by transmitting or storing the parameters of the model and the residual sequence. The encoding can be exact if the required compression is to be lossless, or approximate if the compression can be lossy. ♦

The type of structure or redundancy that existed in these data follows a simple law. Once we recognize this law, we can make use of the structure to *predict* the value of each element in the sequence and then encode the residual. Structure of this type is only one of many types of structure.

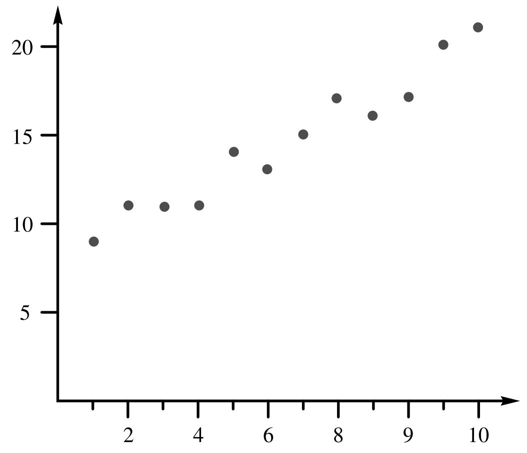


FIGURE 1.2

A sequence of data values.

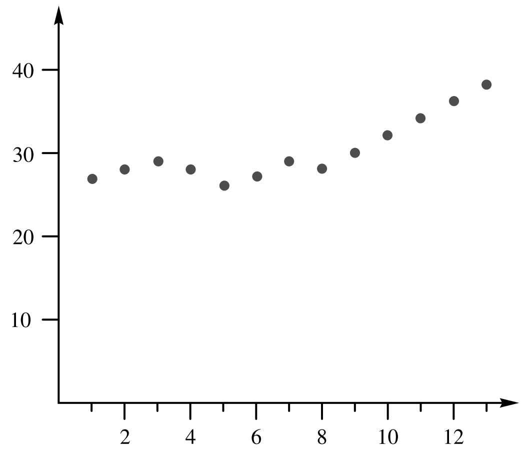


FIGURE 1.3

A sequence of data values.

Example 1.2.2. Consider the following sequence of numbers:

27	28	29	28	26	27	29	28	30	32	34	36	38

The sequence is plotted in Fig. 1.3.

The sequence does not seem to follow a simple law as in the previous case. However, each value in this sequence is close to the previous value. Suppose we send the first value, then in place of subsequent

Table 1.1 A Code With Codewords of Varying Length

a	1
·	001
b	01100
f	0100
n	0111
r	000
w	01101
y	0101

values we send the difference between it and the previous value. The sequence of transmitted values would be

27	1	1	−1	−2	1	2	−1	2	2	2	2	2

Like the previous example, the number of distinct values has been reduced. Fewer bits are required to represent each number, and compression is achieved. The decoder adds each received value to the previous decoded value to obtain the reconstruction corresponding to the received value. Techniques that use the past values of a sequence to *predict* the current value and then encode the error in prediction, or residual, are called *predictive coding* schemes. We will discuss lossless predictive compression schemes in Chapter 7 and lossy predictive coding schemes in Chapter 11.

Assuming both encoder and decoder know the model being used, we would still have to send the value of the first element of the sequence. ♦

A very different type of redundancy is statistical in nature. Often we will encounter sources that generate some symbols more often than others. In these situations, it will be advantageous to assign binary codes of different lengths to different symbols.

Example 1.2.3. Suppose we have the following sequence:

$$a{\cdot}barrayaran{\cdot}array{\cdot}ran{\cdot}far{\cdot}faar{\cdot}faaar{\cdot}away$$

which is typical of all sequences generated by a source (· denotes a blank space). Notice that the sequence is made up of eight different symbols. In order to represent eight symbols, we need to use 3 bits per symbol. Suppose instead we used the code shown in Table 1.1. Notice that we have assigned a codeword with only a single bit to the symbol that occurs most often (a) and correspondingly longer codewords to symbols that occur less often. If we substitute the codes for each symbol, we will use 106 bits to encode the entire sequence. As there are 42 symbols in the sequence, this works out to approximately 2.52 bits per symbol. This means we have obtained a compression ratio of 1.16:1. We will study how to use statistical redundancy of this sort in Chapters 3 and 4. ♦

When dealing with text, along with statistical redundancy, we also see redundancy in the form of words that repeat often. We can take advantage of this form of redundancy by constructing a list of these words and then representing them by their position in the list. This type of compression scheme is called a *dictionary* compression scheme. We will study these schemes in Chapter 5.

Often the structure or redundancy in the data becomes more evident when we look at groups of symbols. We will look at compression schemes that take advantage of this in Chapters 4 and 10.

Finally, there will be situations in which it is easier to take advantage of the structure if we decompose the data into a number of components. We can then study each component separately and use a model appropriate to that component. We will look at such schemes in Chapters 13, 14, and 15.

There are a number of different ways to characterize data. Different characterizations will lead to different compression schemes. We will study these compression schemes in the upcoming chapters and use a number of examples that should help us understand the relationship between the characterization and the compression scheme.

With the increasing use of compression, there has also been an increasing need for standards. Standards allow products developed by different vendors to communicate. Thus, we can compress something with products from one vendor and reconstruct it using the products of a different vendor. The different international standards organizations have responded to this need, and a number of standards for various compression applications have been approved. We will discuss these standards as applications of the various compression techniques.

Finally, compression is still largely an art; and to gain proficiency in an art, you need to get a feel for the process. To help, we have developed software implementations of most of the techniques discussed in this book and have also provided the data sets used for developing the examples in this book. Details on how to obtain these programs and data sets are provided in the Preface. You should use these programs on your favorite data or on the data sets provided in order to understand some of the issues involved in compression. We would also encourage you to write your own software implementations of some of these techniques, as very often the best way to understand how an algorithm works is to implement the algorithm.

1.3 SUMMARY

In this chapter, we have introduced the subject of data compression. We have provided some motivation for why we need data compression and defined some of the terminology used in this book. Additional terminology will be defined as needed. We have briefly introduced the two major types of compression algorithms: lossless compression and lossy compression. Lossless compression is used for applications that require an exact reconstruction of the original data, while lossy compression is used when the user can tolerate some differences between the original and reconstructed representations of the data. An important element in the design of data compression algorithms is the modeling of the data. We have briefly looked at how modeling can help us in obtaining more compact representations of the data. We have described some of the different ways we can view the data in order to model it. The more ways we have of looking at the data, the more successful we will be in developing compression schemes that take full advantage of the structures in the data.

1.4 PROJECTS AND PROBLEMS

1. Use the compression utility on your computer to compress different files. Study the effect of the original file size and type on the ratio of the compressed file size to the original file size.
2. Take a few paragraphs of text from a popular magazine and compress them by removing all words that are not essential for comprehension. For example, in the sentence, "This is the dog that belongs to my friend," we can remove the words *is*, *the*, *that*, and *to* and still convey the same meaning. Let the ratio of the words removed to the total number of words in the original text be the measure of redundancy in the text. Repeat the experiment using paragraphs from a technical journal. Can you make any quantitative statements about the redundancy in the text obtained from different sources?

CHAPTER

MATHEMATICAL PRELIMINARIES FOR LOSSLESS COMPRESSION 2

2.1 OVERVIEW

The treatment of data compression in this book is not very mathematical. (For a more mathematical treatment of some of the topics covered in this book, see [3–6].) However, we do need some mathematical preliminaries to appreciate the compression techniques we will discuss. Compression schemes can be divided into two classes: lossy and lossless. Lossy compression schemes involve the loss of some information and data that have been compressed using a lossy scheme generally cannot be recovered exactly. Lossless schemes compress the data without loss of information, and the original data can be recovered exactly from the compressed data. Notice that we do not use the words "data" and "information" interchangeably. Lossless compression results in a compact representation of the information contained in the data. In order to analyze and develop lossless compression algorithms we need a clear unambiguous quantitative definition of information. This will give us some idea of how much compression can be obtained. If we have a probabilistic description of the data being compressed, Information Theory can provide us with both an unambiguous quantitative definition of information and a bound on how well we can compress the data. We will introduce some basic concepts from information theory in the first part of this chapter. These concepts will help us provide a framework for the development of lossless compression schemes.

Compression algorithms generally consist of two parts: modeling of the data to be compressed and coding with respect to a model. We will review some common approaches to mathematical modeling which we will use in later chapters when we study different compression algorithms. We will also review some basic requirements for codes used in lossless compression. In particular, we will look at a very useful class of codes called prefix codes and show that we can restrict our attention to just this class of codes without losing out on compression performance.

Finally, we briefly look at an alternative approach to understanding information, namely algorithmic information theory, and an alternative framework for compression. We have assumed some knowledge of probability concepts (see Appendix A for a brief review of probability and random processes).

2.2 A BRIEF INTRODUCTION TO INFORMATION THEORY

Although the idea of a quantitative measure of information has been around for a while, the person who pulled everything together into what is now called information theory was Claude Elwood Shannon [3], an electrical engineer and mathematician at Bell Labs. Shannon defined a quantity called *self-information*. Suppose we have an event A, which is a set of outcomes of some random experiment. If $P(A)$ is the probability that the event A will occur, then the self-information associated with A is

Introduction to Data Compression. DOI: 10.1016/B978-0-12-809474-7.00002-1

given by

$$i(A) = \log_b \frac{1}{P(A)} = -\log_b P(A). \tag{2.1}$$

Note that we have not specified the base b of the log function. We will discuss the choice of the base in more detail later in the chapter. The use of the logarithm to obtain a measure of information was not an arbitrary choice as we shall see later in this chapter. But first let's see if the use of a logarithm in this context makes sense from an intuitive point of view. Recall that $\log(1) = 0$, and $-\log(x)$ increases as x decreases from one to zero. Therefore, if the probability of an event is low, the amount of self-information associated with it is high; if the probability of an event is high, the information associated with it is low. Even if we ignore the mathematical definition of information and simply use the definition we use in everyday language, this makes some intuitive sense. The barking of a dog during a burglary is a high-probability event and, therefore, does not contain too much information. However, if the dog did not bark during a burglary, this is a low-probability event and contains a lot of information. (Obviously, Sherlock Holmes understood information theory!)[1] Although this equivalence of the mathematical and semantic definitions of information holds true most of the time, it does not hold all of the time. For example, a totally random string of letters will contain more information (in the mathematical sense) than a well-thought-out treatise on information theory.

Another property of this mathematical definition of information that makes intuitive sense is that the information obtained from the occurrence of two independent events is the sum of the information obtained from the occurrence of the individual events. Suppose A and B are two independent events. The self-information associated with the occurrence of both event A *and* event B is, by Eq. (2.1),

$$i(AB) = \log_b \frac{1}{P(AB)}.$$

As A and B are independent,

$$P(AB) = P(A)P(B)$$

and

$$\begin{aligned} i(AB) &= \log_b \frac{1}{P(A)P(B)} \\ &= \log_b \frac{1}{P(A)} + \log_b \frac{1}{P(B)} \\ &= i(A) + i(B). \end{aligned}$$

The unit of information depends on the base of the log. If we use log base 2, the unit is *bits*; if we use log base e, the unit is *nats*; and if we use log base 10, the unit is *hartleys*. In general, if we do not explicitly specify the base of the log we will be assuming a base of 2.

[1] *Silver Blaze* by Arthur Conan Doyle.

Because the logarithm base 2 probably does not appear on your calculator, let's briefly review logarithms. Recall that

$$\log_b x = a$$

means that

$$b^a = x.$$

Therefore, if we want to take the log base 2 of x

$$\log_2 x = a \Rightarrow 2^a = x,$$

we want to find the value of a. We can take the natural log (log base e) which we will write as *ln*, or log base 10 of both sides (which do appear on your calculator). Then

$$\ln(2^a) = \ln x \Rightarrow a \ln 2 = \ln x$$

and

$$a = \frac{\ln x}{\ln 2}.$$

Example 2.2.1. Let H and T be the outcomes of flipping a coin. If the coin is fair, then

$$P(H) = P(T) = \frac{1}{2}$$

and

$$i(H) = i(T) = 1 \text{ bit.}$$

If the coin is not fair, then we would expect the information associated with each event to be different. Suppose

$$P(H) = \frac{1}{8}, \qquad P(T) = \frac{7}{8}.$$

Then

$$i(H) = 3 \text{ bits}, \qquad i(T) = 0.193 \text{ bits.}$$

At least mathematically, the occurrence of a head conveys much more information than the occurrence of a tail. As we shall see later, this has certain consequences for how the information conveyed by these outcomes should be encoded. ♦

If we have a set of independent events A_i, which are sets of outcomes of some experiment $\mathcal{S}$, such that

$$\bigcup A_i = S$$

Table 2.1 The Most Probable Five Sequences of Lengths 1, 2, 3 and 10 From *Peter Pan* by J.M. Barrie, *The Communist Manifesto* by K. Marx and F. Engels, and *The Wealth of Nations* by A. Smith. (All Text Files Obtained From the *Gutenberg Project*)

$n = 1$			$n = 2$			$n = 3$			$n = 10$		
·	·	·	*e*·	*e*·	*e*·	·*th*	·*th*	·*th*	*they*·*were*	·*bourgeois*	*therefore,*
e	*e*	*e*	*he*	*th*	·*t*	*the*	*the*	*the*	.·*darling*·	*bourgeoisi*	·*which*·*the*
t	*t*	*t*	·*t*	·*t*	*th*	*he*·	*he*·	*he*·	,”·*he*·*said*	*ourgeoisie*	·*the*·*great*
a	*o*	*o*	*th*	*he*	*he*	*and*	·*of*	·*of*	·*they*·*were*	*proletaria*	·*the*·*same*
h	*i*	*a*	*d*·	*s*·	·*a*	*nd*·	*of*·	*of*·	*mrs.*·*darli*	*ebourgeoi*	*herefore,*·

where S is the sample space, then the average self-information associated with the random experiment is given by

$$H = \sum P(A_i)i(A_i) = -\sum P(A_i)\log_b P(A_i).$$

This quantity is called the *entropy* associated with the experiment. One of the many contributions of Shannon was that he showed that if the experiment is a source that puts out symbols A_i from a set $\mathcal{A}$, then the entropy is a measure of the average number of binary symbols needed to code the output of the source. Shannon showed that the best that a lossless compression scheme can do is to encode the output of a source with an average number of bits equal to the entropy of the source.

The set of symbols $\mathcal{A}$ is often called the *alphabet* for the source, and the symbols are referred to as *letters*. In our definition of entropy we have assumed that a general source $\mathcal{S}$ with alphabet $\mathcal{A} = \{1, 2, \ldots, m\}$ that generates a sequence $\{X_1, X_2, \ldots\}$, the elements in the sequence will be generated independently. Thus each letter appears as a surprise. In practice this is not necessarily the case and there may be considerable dependence between letters. These dependencies will effect the entropy of the source. In later sections we will look at specific ways to model these dependencies for various sources of interest. However, in order to make a general statement about the effect of these dependencies on the entropy of stationary sources we need a general approach which will capture all dependencies. One way to capture dependencies is to look at the joint distributions of longer and longer sequences generated by the source. Consider the n-length most likely sequences from three very different texts shown in Table 2.1 for $n = 1, 2, 3, 4$. We can see that for n small, all we get is the inherent structure of the English language. However, as we increase n to 10 we can identify the particular text simply by looking at the five most probable sequences. That is, as we increase n we capture more and more of the structure of the sequence. Define G_n as

$$G_n = -\sum_{i_1=1}^{i_1=m}\sum_{i_2=1}^{i_2=m}\cdots\sum_{i_n=1}^{i_n=m} P(X_1 = i_1, X_2 = i_2, \ldots, X_n = i_n)\log P(X_1 = i_1, X_2 = i_2, \ldots, X_n = i_n).$$

Then this quantity will denote the amount of information contained in n-tuples from the source. The per-letter information can be obtained by normalizing G_n as

$$H_n = \frac{1}{n}G_n.$$

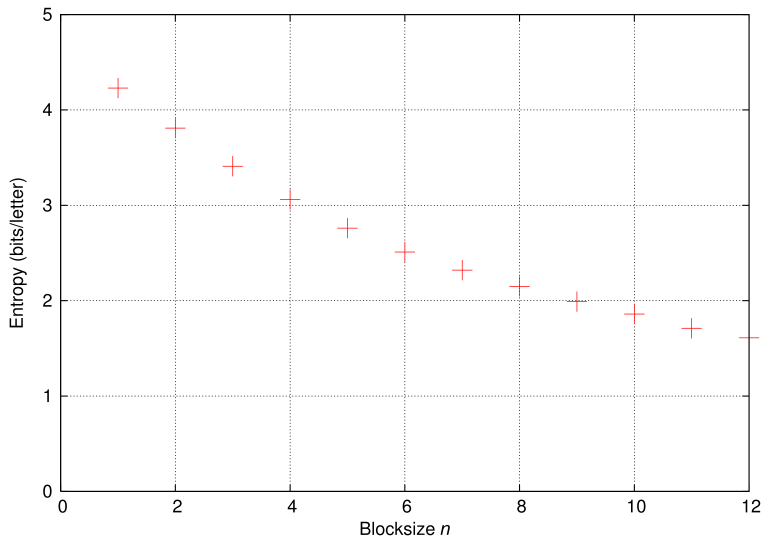


FIGURE 2.1

H_n in bits per letter for $n = 1, \ldots, 12$ for *Wealth of Nations*.

If we plot this quantity for n from 1 to 12 for the book *Wealth of Nations* we obtain the values shown in Fig. 2.1. We can see that H_n is converging to a particular value. Shannon showed [3] that for a stationary source, in the limit this value will converge to the entropy.

$$H(\mathcal{S}) = \lim_{n \to \infty} H_n. \tag{2.2}$$

If each element in the sequence is independent and identically distributed (*iid*), then we can show that

$$G_n = -n \sum_{i_1=1}^{i_1=m} P(X_1 = i_1) \log P(X_1 = i_1) \tag{2.3}$$

and the equation for the entropy becomes

$$H(\mathcal{S}) = -\sum P(X_1) \log P(X_1). \tag{2.4}$$

For most sources, Eqs. (2.2) and (2.4) are not identical. If we need to distinguish between the two, we will call the quantity computed in (2.4) the *first-order entropy* of the source, while the quantity in (2.2) will be referred to as the *entropy* of the source.

In general, it is not possible to know the entropy for a physical source, so we have to estimate the entropy. The estimate of the entropy depends on our assumptions about the structure of the source sequence.

Consider the following sequence:

$$1\ 2\ 3\ 2\ 3\ 4\ 5\ 4\ 5\ 6\ 7\ 8\ 9\ 8\ 9\ 10$$

Assuming the frequency of occurrence of each number is reflected accurately in the number of times it appears in the sequence, we can estimate the probability of occurrence of each symbol as follows:

$$P(1) = P(6) = P(7) = P(10) = \frac{1}{16}$$
$$P(2) = P(3) = P(4) = P(5) = P(8) = P(9) = \frac{2}{16}.$$

Assuming the sequence is *iid*, the entropy for this sequence is the same as the first-order entropy as defined in (2.4). The entropy can then be calculated as

$$H = -\sum_{i=1}^{10} P(i) \log_2 P(i).$$

With our stated assumptions, the entropy for this source is 3.25 bits. This means that the best scheme we could find for coding this sequence could only code it at 3.25 bits/sample.

However, if we assume that there was sample-to-sample correlation between the samples and we remove the correlation by taking differences of neighboring sample values, we arrive at the *residual* sequence

$$1\ 1\ 1\ -1\ 1\ 1\ 1\ \ -1\ 1\ 1\ 1\ 1\ 1\ \ -1\ 1\ 1$$

This sequence is constructed using only two values with probabilities: $P(1) = \frac{13}{16}$ and $P(-1) = \frac{3}{16}$. The entropy in this case is 0.70 bits per symbol. Of course, knowing only this sequence would not be enough for the receiver to reconstruct the original sequence. The receiver must also know the process by which this sequence was generated from the original sequence. The process depends on our assumptions about the structure of the sequence. These assumptions are called the *model* for the sequence. In this case, the model for the sequence is

$$x_n = x_{n-1} + r_n$$

where x_n is the nth element of the original sequence and r_n is the nth element of the residual sequence. This model is called a *static* model because its parameters do not change with n. A model whose parameters change or adapt with n to the changing characteristics of the data is called an *adaptive* model.

Basically, we see that knowing something about the structure of the data can help to "reduce the entropy." We have put "reduce the entropy" in quotes because the entropy of the source is a measure of the amount of information generated by the source. As long as the information generated by the source is preserved (in whatever representation), the entropy remains the same. What we are reducing is our estimate of the entropy. The "actual" structure of the data in practice is generally unknowable, but anything we can learn about the data can help us to estimate the actual source entropy. Theoretically, as seen in Eq. (2.2), we accomplish this in our definition of the entropy by picking larger and larger blocks of data to calculate the probability over, letting the size of the block go to infinity.

Consider the following contrived sequence:

$$1\ 2\ 1\ 2\ 3\ 3\ 3\ 3\ 1\ 2\ 3\ 3\ 3\ 3\ 1\ 2\ 3\ 3\ 1\ 2$$

Obviously, there is some structure to this data. However, if we look at it one symbol at a time, the structure is difficult to extract. Consider the probabilities: $P(1) = P(2) = \frac{1}{4}$, and $P(3) = \frac{1}{2}$. The entropy is 1.5 bits/symbol. This particular sequence consists of 20 symbols; therefore, the total number of bits required to represent this sequence is 30. Now let's take the same sequence and look at it in blocks of two. Obviously, there are only two symbols, 1 2, and 3 3. The probabilities are $P(1\ 2) = \frac{1}{2}$, $P(3\ 3) = \frac{1}{2}$, and the entropy is 1 bit/symbol. As there are 10 such symbols in the sequence, we need a total of 10 bits to represent the entire sequence—a reduction of a factor of three. The theory says we can always extract the structure of the data by taking larger and larger block sizes; in practice, there are limitations to this approach. To avoid these limitations, we try to obtain an accurate model for the data and code the source with respect to the model. In Section 2.3, we describe some of the models commonly used in lossless compression algorithms. But before we do that, let's make a slight detour and see a more rigorous development of the expression for average information. While the explanation is interesting, it is not really necessary for understanding much of what we will study in this book and can be skipped.

2.2.1 DERIVATION OF AVERAGE INFORMATION ★

We start with the properties we want in our measure of average information. We will then show that requiring these properties in the information measure leads inexorably to the particular definition of average information, or entropy, that we have provided earlier.

Given a set of independent events $A_1, A_2, \ldots, A_n$ with probability $p_i = P(A_i)$, we desire the following properties in the measure of average information H:

1. We want H to be a continuous function of the probabilities p_i. That is, a small change in p_i should only cause a small change in the average information.
2. If all events are equally likely, that is, $p_i = 1/n$ for all i, then H should be a monotonically increasing function of n. The more possible outcomes there are, the more information should be contained in the occurrence of any particular outcome.
3. Suppose we divide the possible outcomes into a number of groups. We indicate the occurrence of a particular event by first indicating the group it belongs to, then indicating which particular member of the group it is. Thus, we get some information first by knowing which group the event belongs to; and then we get additional information by learning which particular event (from the events in the group) has occurred. The information associated with indicating the outcome in multiple stages should not be any different than the information associated with indicating the outcome in a single stage.

   For example, suppose we have an experiment with three outcomes, A_1, A_2, and A_3, with corresponding probabilities, p_1, p_2, and p_3. The average information associated with this experiment is simply a function of the probabilities:

$$H = H(p_1, p_2, p_3).$$