**Introduction to data compression**

In the last decade, we have been witnessing a transformation—some call it a revolution—in the way we communicate, and the process is still underway. 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 on your MP3 player or watch a DVD, and you are being entertained courtesy of compression. So what is data compression, and why do we need it? Most of you have heard of JPEG and MPEG, which are standards for representing images, video, and audio. Data compression algorithms are used in these standards to reduce the number of bits required to represent an image or a video sequence or music. In brief, 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 18), 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 (44,100samples 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 it is fully operational. New sequencing technology is resulting in ever-increasing database sizes containing genomic information while new medical scanning technologies could 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 transmitted without using compression, including CD-ROMs, optical fibers, Asymmetric Digital Subscriber Lines (ADSL), and cable modems. 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, 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* (·) and *a* (**·−**), and longer sequences to letters that occur less frequently, such as *q* (**−−·−**) and *j* (**·−−−**).

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. 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 , 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%.

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. That is, 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 currently being used 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.

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 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.

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 *Y* and the original *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.

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.

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, if we assume 8 bits per byte (or pixel), 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.

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.