Data Compression

2019/2020

“Learn why data compression has become crucial as data production continues to skyrocket.”

Instructor: Dr. Ali Kadhum M. Al-Quraby

College Web Site	http://staff.uobabylon.edu.iq/lectures.aspx?id=41179
Google Classroom	https://classroom.google.com/c/MzcyNDY3MjU4NzBa

"Compression is all about the most compact representation of data."

Introduction

The last decade we have been witnessing a transformation—some call it a revolution—in the way we communicate, and the process is still under way. This transformation includes the 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.

Data compression addresses the problem of **reducing** the amount of **data required** to represent a digital file, so that it can be stored or transmitted so efficiently.

The principle of data compression is that, it compresses data by **removing redundancy** from the original data in the source file.

On the other hand**,** ***information theory*** tells us that the **amount of information** conveyed by an event **relates** to its **probability of occurrence**. An event that is less likely to occur is said to contain more information than an event that is more likely to occur. The amount of information of an event and its probability are thus **opposite**.

Amount of information ↓ **probability** ↑ **compression** ↑ **probability** ↑

It is obvious that information theory is the base theory that data compression relies on.

The problem of **representing** the source alphabet symbols Si in term of another system of symbols (usually the binary system consisting of the two symbols 0 & 1) is the main topic of ***coding theory***.

An optimum coding scheme will use **more bits** for the symbols that less likely to occur, and a **fewer bits** for the symbols that frequently occur.

Logically speaking, coding theory leads to information theory and information theory provides the bounds on what can be done by suitable encoding of the information. Thus, the two theories are intimately related.

Before delivering into the details, we discuss important data compression terms:

Data Compression Terminology

Data Compression

is the process of converting an input data stream (the source stream or the original raw data) into another data stream (the output, or the compressed, stream) that has a smaller size. A stream is either a file or a buffer in memory. Data compression is popular because of two reasons:

1. People like to accumulate data and hate to throw anything away. No matter how big a storage device one has, sooner or later it is going to overflow. Data compression seems useful because it delays this inevitability.

2. People hate to wait a long time for data transfers. When sitting at the computer, waiting for a Web page to come in, or for a file to download, we naturally feel that anything longer than a few seconds is a long time to wait.

There are many known methods for data compression. They are based on different ideas, are suitable for different types of data, and produce different results, but they are all based on the **same principle**, namely, they compress data by removing ***redundancy*** from the original data in the source file. Any nonrandom collection data has some structure, and this structure can be exploited to achieve a smaller representation of the data, a representation where no structure is discernible.

The idea of compression by reducing redundancy suggests the **general law** of data compression, which is to "assign short codes to common events (symbols or phrases) and long codes to rare events." There are many ways to implement this law, and an analysis of any compression method shows that, deep inside, it works by obeying the general law.

Type of Redundancy

1. ***Text redundancy***:
   - In typical English text, for example, the letter E appears very often, while Z is rare. This is called ***alphabetic redundancy***, and suggests assigning variable-size codes to the letters, with E getting the shortest code and Z, the longest one.
   - Another type of redundancy, ***contextual redundancy***, is illustrated by the fact that the letter Q is almost always followed by the letter U (i.e., that certain diagrams and trigrams are more common in plain English than others).
2. ***Images redundancy***: is illustrated by the fact that in a nonrandom image, **adjacent pixels** tend to have **similar colors**. The primary types of redundancy can be found in image:
   - **Coding**: Coding redundancy occurs when the data used to represent the image not utilized in an optimal manner. For example, if we have an 8 bits/ pixel image that allows 256 gray- level value, but the actual image contains only 16 gray-level values, this is a suboptimal coding, where only 4 bits/pixel are actually needed.
   - **Interpixel** (Spatial Redundancy): Interpixel redundancy occurs because adjacent pixels tend to be highly correlated. This is a result of the fact that in most images the brightness levels do not change rapidly, but change gradually, so that adjacent pixel values tend to be relatively close to each other in value (for video, or motion images, this concept can be extended to include interframe redundancy, redundancy between frames of image data).
   - Psychovisual redundancy: refers to the fact that some information is more important to the human visual system than other types of information. For example, we can only perceive spatial frequencies below about 50 cycles per degree so that any higher-frequency information is of little interest to us.
3. ***Video redundancy***: is illustrated by the fact that in a nonrandom video **consecutive frames** tend to be **similar.**

The principle of compressing by removing redundancy also answers the following ***question***: "***Why is it that an already compressed file cannot be compressed further?***" The answer, of course, is that such a file ***has little or no redundancy***, so there

is nothing to remove. An example of such a file is random text. When such a file is compressed, there is no redundancy to remove. If we assume that there was a possibility to compress an already compressed file, then successive compressions would reduce the size of the file until it becomes a single byte, or even a single bit. This, of course, is ridiculous since a single byte cannot contain the information present in an arbitrarily large file.

Data compression is achieved by reducing redundancy, but this also makes the data **less reliable**, more prone to errors. Making data more reliable, on the other hand, is done by adding ***check bits and parity bits***, a process that increases the size of the codes, thereby increasing redundancy. Data compression and ***data reliability*** are thus opposites. Before delving into the details, we discuss important data compression terms.

- The ***compressor*** or *encoder* is the **program** that compresses the raw data in the input stream and creates an output stream with compressed (low redundancy) data. The ***decompressor*** or *decoder* converts in the opposite direction.
- The term "***stream***" is used throughout this lecture **instead of** "file". "Stream" is a more general term because the compressed data may be **transmitted directly** to the decoder, instead of being written to a file and saved. Also, the data to be compressed may be **downloaded** from a network instead of being input from a file.
- For the original input stream, we use the terms *un encoded,* ***raw data***. The contents of the final, compressed, stream is considered the *encoded* or ***compressed*** data. The term ***bit stream*** is also used in the literature to indicate the compressed stream.

Type of Data Compression

- A ***non-adaptive*** compression method is **rigid** and does **not modify** its operations, its parameters, or its tables in response to the particular data being compressed. Such a method is best used to compress data that is all of a single type. Examples are the Group 3 and Group 4 methods for facsimile compression. They are specifically designed for facsimile compression and would do a poor job compressing any other data. In contrast, an ***adaptive*** method examines the raw data and **modifies** its operations and/or its parameters accordingly. An example is the adaptive Huffman

method. Some compression methods use a 2-pass algorithm, where the first pass reads the input stream to collect statistics on the data to be compressed, and the second pass does the actual compressing using parameters set by the first pass. Such a method may be called ***semi adaptive***.

- ***Lossy / lossless compression****:* Certain compression methods are ***lossy.*** They achieve better compression by losing some information. When the compressed stream is decompressed, the result is not identical to the original data stream. Such a method makes sense especially in compressing images, movies, or sounds. If the loss of data is small, we may not be able to tell the difference. In contrast, text files, especially files containing computer programs, may become worthless if even one bit gets modified. Such files should be compressed only by a ***lossless*** compression method, also special purpose images like medical images, forensic images, NASA images are compressed using ***lossless*** compression methods.
- ***Symmetrical*** *compression* is the case where the compressor and decompressor use basically the same algorithm but work in "opposite" directions. Such a method makes sense for general work, where the same number of files are compressed as are decompressed. In an ***asymmetric*** compression method either the compressor or the decompressor may have to work significantly harder (i.e. each one uses a different algorithm).

Benefits of Data Compression

The digital representation of the data usually required a very large number of bits. In many applications, it is important to consider techniques for representing data with fewer bits, while maintaining an acceptable fidelity of data quality.

The main benefits of data compression are the follows:

1. Reducing the **storage** requirement or saving the storage space.
2. Potential **cost** saving associated with sending less data over communication channels (e.g. the cost of call is usually depending on its duration).
3. Compression can reduce the probability of **transmission error** occurring since fewer characters are transmitted when data is compressed.

4. By converting the original data that is represented by conventional code into a different (compressed) code, compression algorithms may provide a **level of security**.
5. Reducing the time required for transmission of the total original image by transmitting its compressed version.

Compression Performance

Most compression methods are ***physical***. They look only at the bits in the input stream and ignore the meaning of the data items in the input (e.g., the data items may be words, pixels, or sounds). Such a method translates one bit stream into another, shorter, one.

The performance of a compression algorithm can be measured by various criteria. It depends on what is our priority concern. 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, and how closely the reconstruction resembles the original.

A very logical way of measuring how well a compression algorithm is to compresses a given set of data and look at the difference in size of the data before the compression and size of the data after the compression.

There are several ways of measuring the compression effect:

✓ Compression Ratio

This is simply the ratio of size after compression to size before compression. Values greater than 1 imply an output stream bigger than the input stream (negative compression). The compression ratio can also be called bpb (bit per bit), The compression factor is denoted by:

$$\textbf{\textit{Compression Ratio}} = \frac{\textbf{\textit{size of the output stream}}}{\textbf{\textit{size of the input stream}}}$$

✓ Compression Factor

Data compression involve reducing the size of data file, while retaining necessary information. The reduced file is called the compressed file and is used to reconstruct the

original file, resulting in the decompressed file. The original file, before any compression is performed is called the uncompressed file. The ratio of the original, uncompressed file and the compressed file is referred to as the ***compression factor***. This is the reverse of compression ratio. The compression factor is denoted by:

$$Compression\ Factor = \frac{size\ of\ the\ input\ stream}{size\ of\ the\ output\ stream}$$

In this case value greater than 1 indicates compression, and values less than 1 imply expansion. This measure seems natural to many people, since the bigger the factor, the better the compression.

- ✓ **Saving Percentage:** This shows the shrinkage as a percentage

$$Saving\ Percentage = \frac{size\ of\ the\ input\ stream - size\ of\ the\ output\ stream}{size\ of\ the\ input\ stream}\%$$

- ✓ **Bit Per Pixel**

Another way to state the compression of an image is to use the terminology of ***bit per pixel***. For an N×N image.

$$Bit\ per\ pixel = \frac{Number\ of\ Bits(compressed\ file)}{Number\ of\ Pixels(original\ file)} = \frac{8\times(Number\ of\ Bytes)}{N\times N}$$

Example 1:

The original image is 256 × 256 pixel. Single-band (gray scale) 8 bits per pixel. This file size is 65,536 bytes (64 k). After compression the image file size is became 6,554 bytes. Compute the compression ratio, the compression factor, the saving percentage and the Bit per pixel.

$$CR = \frac{6554}{65536} = 0.1$$

A value of **0.1** means that the data occupies **10%** of its original size after compression.

$$CF = \frac{65536}{6554} = 9.99 \approx 10$$

This can also be written as 10:1. This is called "10 to 1 compression" or a "10 times compression" or it can be stated as "compressing the image to 1/10 its original size".

$$SP = \frac{65536-6554}{65536} \times 100 = 0.8999 \times 100 \approx 90\ \%$$

A value of **90** means that the output stream occupies **10%** of its original size (or that the compression has resulted in savings of **90%**).

Using the above example, with a compression factor of **65,536/6,554 bytes**, we want to express this as ***bits per pixel***. This is done by first finding the number of pixels in the image = 256×256=65,536 pixels. We then find the number of bits in the compressed image file = (6,554 bytes) (8 bits/bytes) = 52,432 bits. Now we can find the bits per pixel by taking the ratio

$$BPP = \frac{52432}{65536} = 0.8 \ \ bits/pixel$$

Example 2:

The original image is 256 × 256 pixel. Single-band (gray scale) 8 bits per pixel. This file size is 65,536 bytes (64 k). After compression the image file size is became 16,384 bytes. Compute the compression ratio, the compression factor, the saving percentage and the Bit per pixel.

The reduction in the file size is necessary to meet the **bandwidth requirement** for many transmission systems, as well as the storage requirement in computer data bases. The amount of data required for digital images is enormous.

For example, a single 512 × 512, 8-bit image required 2,097,152 bits for storage. If we wanted to **transmit** this image over the World Wide Web, it would probably take minutes for transmission- too long for most people to wait.

Example 3

To transmit an RGB (true color) 512 × 512, 24-bit (8 bit / color) image via modem at 28.8 kbaud (kilobits/second), it would take about:

$$Time = \frac{Total\ size\ of\ file}{Speed\ of\ transmitted\ Data} = \frac{(512 \times 512\ pixel) \times (24\ bits/pixel)}{(28.8 \times 1024\ bits/second)} = 213\ seconds = 3.6\ Minutes$$

Example 4

A colored video clip of 4 second duration with a frame size of 160×120 pixels and a frame rate of 30 frames per second, is to be transmitted via modem at 28.8 kbaud (kilobits/second), it would take about:

$$\boldsymbol{Time = \frac{(160 \times 120\ pixel) \times (24\ bits/pixel) \times (4\ seconds) \times (30\ frame/second)}{(28.8 \times 1024\ bits/second)} = 1875\ seconds = 31.25\ Minutes}$$

The above results show the necessity of data compression especially in images and movies transmission.