Coding and Data Compression

Lec. 1

Introduction

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, the bitstream, or the compressed stream) that has a smaller size. A stream can be a file, a buffer in memory, or individual bits sent on a communications channel.

The decades of the 1980s and 1990s saw an exponential decrease in the cost of digital storage. There seems to be no need to compress data when it can be stored inexpensively in its raw format, yet the same two decades have also experienced rapid progress in the development and applications of data compression techniques and algorithms. The following paragraphs try to explain this apparent paradox.

- Many like to accumulate data and hate to throw anything away. No matter how big a storage device one has. Data compression is useful for this.
- As storage devices get bigger and cheaper, it becomes possible to create, store, and transmit larger and larger data files. In the old days of computing, most files were text or executable programs and were therefore small. No one tried to create and process other types of data simply because there was no room in the computer. In the 1970s, with the advent of semiconductor memories and floppy disks, still images, which require bigger files, became popular. These were followed by audio and video files, which require even bigger files.
- We hate to wait 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. Compressing data before it is transmitted is therefore a natural solution.
- CPU speeds and storage capacities have increased dramatically in the last two decades, but the speed of mechanical components (and therefore the speed of

disk input/output) has increased by a much smaller factor. Thus, it makes sense to store data in compressed form, even if plenty of storage space is still available on a disk drive. Compare the following scenarios: (1) A large program resides on a disk. It is read into memory and is executed. (2) The same program is stored on the disk in compressed form. It is read into memory, decompressed, and executed. It may come as a surprise to learn that the latter case is faster in spite of the extra CPU work involved in decompressing the program. This is because of the huge disparity between the speeds of the CPU and the mechanical components of the disk drive.

- A similar situation exists with regard to digital communications. Speeds of communications channels, both wired and wireless, are increasing steadily but not dramatically. It therefore makes sense to compress data sent on telephone lines between fax machines, data sent between cellular telephones, and data (such as web pages and television signals) sent to and from satellites.

Advantages/Disadvantages of Compression

Compression of files offer many advantages. When compressed, the quantity of bits used to store the information is reduced. Files that are smaller in size will result in shorter transmission times when they are transferred on the Internet. Compressed files also take up less storage space. File compression can zip up several small files into a single file for more convenient email transmission.

As compression is a mathematically intense process, it may be a time consuming process, especially when there is a large number of files involved. Some compression algorithms also offer varying levels of compression, with the higher levels achieving a smaller file size but taking up an even longer amount of compression time. It is a system intensive process that takes up valuable resources that can sometimes result in "Out of Memory" errors. With so many compression algorithm variants, a user downloading a compressed file may not have the necessary program to un-compress it.

Some transmission protocols may include optional compression built-in (e.g. FTP has a MODE-Z compression option), so that taking time to compress data by another process before transmission may negate some of the advantages of using such an option in the protocol (because what is eventually submitted for transmission to/by the protocol is probably now not very further-compressible at all, and may waste time while the protocol tries and fails to achieve more compression). It is distinctly possible that 'external' compression beforehand is more efficient these days, and that any compression option in the protocol should probably be deprecated. However, it is not beyond the bounds of possibility that the built-in compression actually achieves faster overall results, but possibly with larger compressed files, or vice versa. Experimentation should be employed to ascertain which applies, versus which factor is most important to the user.

The field of data compression is often called ***source coding***. We imagine that the input symbols (such as bits, ASCII codes, bytes, audio samples, or pixel values) are emitted by a certain information source and have to be coded before being sent to their destination.

Data compression has come of age in the last 20 years. However, the need for compressing data has been felt in the past, even before the advent of computers, as the following quotation suggests:

I have made this letter longer than usual
because I lack the time to make it shorter.
—Blaise Pascal

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 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 terms *redundancy* and *structure* are used in the professional literature, as well as *smoothness*, *coherence*, and *correlation*; they all refer to the same thing. Thus, redundancy is a key concept in any discussion of data compression.