DATA COMPRESSION

CIT 413

F.J.Babakano

Data Compression

- *Data compression is the science (and art) of representing information in a* compact form.
- It has grown to be now ubiquitous.
- One of the critical enabling technologies for the on-going digital multimedia revolution for decades.
- Applied in ever-growing Internet, digital TV, mobile communication or increasing video communication

Data Compression Cntd.

- It involves designing efficient algorithms to:
  - *represent data in a less redundant fashion*
  - *remove the redundancy in data*
  - *implement coding, including both encoding and decoding.*
- The key approaches of data compression can be summarised as *modelling* + *coding*.
- Modelling is a process of constructing a knowledge system for performing compression.
- Coding includes the design of the code and product of the compact data form.

Aims and objectives

- The subject aims to introduce you to the main issues in data compression
  - common compression techniques for text, audio, image and video data and
  - show you the significance of some compression technologies.
- The objectives of the subject are to:
  - *outline important issues in data compression*
  - *describe a variety of data compression techniques*
  - *explain the techniques for compression of binary programmes, data,* sound and image

Importance of Data Compression

- Data compression techniques is motivated mainly by the need to improve efficiency of information processing.
  - storage efficiency
  - *efficient usage of transmission bandwidth*
  - *reduction of transmission time.*
- There are cases in which extra storage or extra bandwidth is difficult to achieve, if not impossible.
- Data compression as a means may make much more efficient use of existing resources with less cost.

Brief history

- Data compression can be viewed as the art of creating shorthand representations for the data even today, but this process started as early as 1,000 BC.
  - *1000BC, Shorthand*
  - *1829, Braille code*
  - *1843, Morse code*
  - *1930 onwards, Analog compression*
  - *1950, Huffman codes*
  - *1975, Arithmetic coding*
  - *1977, Dictionary-based compression*

Brief history Contd.

- 1980s
  - early 80s, FAX
  - mid-80s, Video conferencing, still images (JPEG), improved FAX standard (JBIG)
  - late 80s, onward Motion video compression (MPEG)
- *1990s*
  - early 90s, Disk compression (stacker)
  - mid-90s, Satellite TV
  - late 90s, Digital TV (HDTV), DVD, MP3

Source data

- The word *data includes any digital information that can* be processed in a computer, which includes text, voice, video, still images, audio and movies.
- The data before any compression (i.e. encoding) process is called the *source data, or the source for short.*
- Three common types of source data in the computer are *text and (digital) image and sound.*
  - *Text data is usually represented by ASCII code (or EBCDIC).*
  - *Image data is represented often by a two-dimensional array of pixels in* which each pixel is associated with its color code.
  - *Sound data is represented by a wave (periodic) function.*
- In the application world, the source data to be compressed is likely to be so-called *multimedia and can be a mixture of text, image and sound.*

Lossless and lossy data compression

- Data compression is simply a means for efficient digital representation of a source of data such as text, image and the sound.
- The goal of data compression is to represent a source in digital form with as few bits as possible while meeting the minimum requirement of reconstruction.
- This goal is achieved by removing any redundancy presented in the source.
- There are two major families of compression techniques in terms of the possibility of reconstructing the original source. They are called *Lossless and lossy compression.*

Lossless compression

- A compression approach is lossless only if it is possible to exactly reconstruct the original data from the compressed version.
- There is no loss of any information during the compression process.
- They are mostly applied to symbolic data such as character text, numeric data, computer source code and executable graphics and icons.
- Lossless compression techniques are also used when the original data of a source are so important that we cannot afford to lose any details. For example medical images, text and images preserved for legal reasons; some computer executable files, etc.

Lossy compression

- A compression method is lossy compression only if it is not possible to reconstruct the original exactly from the compressed version.
- There are some insignificant details that may get lost during the process of compression.
- Approximate reconstruction may be very good in terms of the compression-ratio but usually it often requires a trade-off between the visual quality and the computation complexity (i.e. speed).
- Data such as multimedia images, video and audio are more easily compressed by this method.

Main compression techniques

- Data compression is often called *coding due to the fact that its aim is to find* a specific *short (or shorter) way of representing data.*
- *Encoding and decoding* are used to mean compression and decompression respectively. We outline some major compression algorithms below:
  - *Run-length coding*
  - *Quantisation*
  - *Statistical coding*
  - *Dictionary-based coding*
  - *Transform-based coding*
  - *Motion prediction.*

Run-length coding

- This replaces consecutively repeated symbols in a source with a code pair which consists of either the repeating symbol and the number of its occurrences, or sequence of non-repeating symbols.
- Example: *String ABBBBBBBCC can be represented by* Ar_7Br_2C*, where* r_7 *and* r_2 *means 7 and 2 occurrences respectively*.
- All the symbols are represented by an 8-bit ASCII codeword.

Quantisation

- The basic idea of quantisation is to apply a certain computation to a set of data in order to achieve an approximation in a simpler form.
- Example: *Consider storing a set of integers (7, 223, 15, 28, 64, 37,145). Let x be an integer in the set. We have* $7 <= x <= 223$*. Since* $0 < x < 255$ *and* $2^8 = 256$*, it needs 8 binary bits to represent each integer above.*
- *However, if we use a multiple, say 16, as a common divider to apply to each integer and round its value to the nearest integer, the above set becomes (0,14, 1, 2, 4, 2, 9) after applying the computation x div 16. Now each integer can be stored in 4 bits, since the maximum number 14 is less than* $2^4 = 16$*.*

Statistical coding

- The idea of statistical coding is to use statistical information to replace a fixed-size code of symbols by a, hopefully, shorter variable-sized code.
- Example 1.3 *We can code the more frequently occurring symbols with fewer bits. The statistical information can be obtained by simply counting the frequency of each character in a file. Alternatively, we can simply use the probability of each character.*

Dictionary-based coding

The dictionary approach consists of the following main steps:

1. read the file
2. find the frequently occurring sequences of symbols (FOSSs)
3. build up a dictionary of these FOSSs
4. associate each sequence with an index (usually a fixed length code)
5. replace the FOSS occurrences with the indices.

TRANSFORM-BASED CODING

- The transform-based approach models data by mathematical functions,usually by periodic functions such as *cos(x) and applies mathematical rules* to primarily diffuse data. The idea is to change a mathematical quantity such

  as a sequence of numbers to another form with useful features. It is used mainly in lossy compression algorithms involving the following activities:

Compression performance

$$\textit{Compression Ratio} = \frac{\text{Size After Compression}}{\text{Size before Compression}}$$

$$\textit{Compression Factor} = \frac{\text{Size before Compression}}{\text{Size After Compression}}$$

$$\textit{Compression Percentage} = \frac{\text{Size beforeCompression} - \text{Size After Compression}}{\text{Size before Compression}} \text{ X } 100$$

Example

- *A source image file (pixels 256 X 256) with 65,536 bytes is compressed into a file with 16,384 bytes. The compression ratio is 1/4 and the compression factor is 4. The saving percentage is: 75%*

- *Exercise:* Suppose the size of myfile.gz is 20 KB while the original file myfile is of size 40 KB. Compute the compression ratio, compression factor and saving percentage.

HDC Run Length Encoding

- Example 2.2
  GGG⊔⊔⊔⊔⊔⊔BCDEFG⊔⊔55GHJK⊔LM7777777 77777 *can be compressed to*
  $r_3Gr_6n_6BCDEFGr_2n_955\sqcup LMr_{12}7$
- explain under what conditions a Run-length algorithm may work effectively.
- explain, with an example, how the HDC algorithm works.

Solution

- The first 3 Gs are read and encoded by r_3G.
- The next 6 spaces are found and encoded by r_6.
- The non-repeating symbols BCDEFG are found and encoded by $n_6BCDEFG$.
- The next 2 spaces are found and encoded by r_2.
- The next 9 non-repeating symbols are found and encoded by $n_9 55GHJK\sqcup LM$.
- The next 12 ‘7’s are found and encoded by $r_{12}7$.

Exercise

- Apply the HDC (Hardware Data Compression) algorithm to the following sequence of symbols: kkkk⊔⊔⊔⊔⊔⊔⊔⊔⊔⊔⊔⊔g⊔⊔hh5522777666abbbbc mmj⊔⊔##
- Provide an example of a source file on which the HDC algorithm would perform very badly.

Huffman Encoding

- Huffman coding is a successful compression method used originally for text compression. It assumes that each character is stored as a 8-bit ASCII code.
- Example 3.3 *Compress 'BILL BEATS BEN.' (15 characters in total) using the Huffman approach.*
- Decode it

Modeling and Coding

- data compression essentially consists of two types of work:
- modelling : the embodiment of what the compression algorithm knows about the source domain. Every compression algorithm has to make use of some knowledge about its platform.
- Example 4.1 *Consider the Huffman encoding algorithm. The model is based on the probability distribution of characters of a source text.*

Modelling and Coding

- The device that is used to fulfil the task of coding is usually called *coder* meaning *encoder*. *Based on the model and some calculations, the coder* is used to

  -derive a code

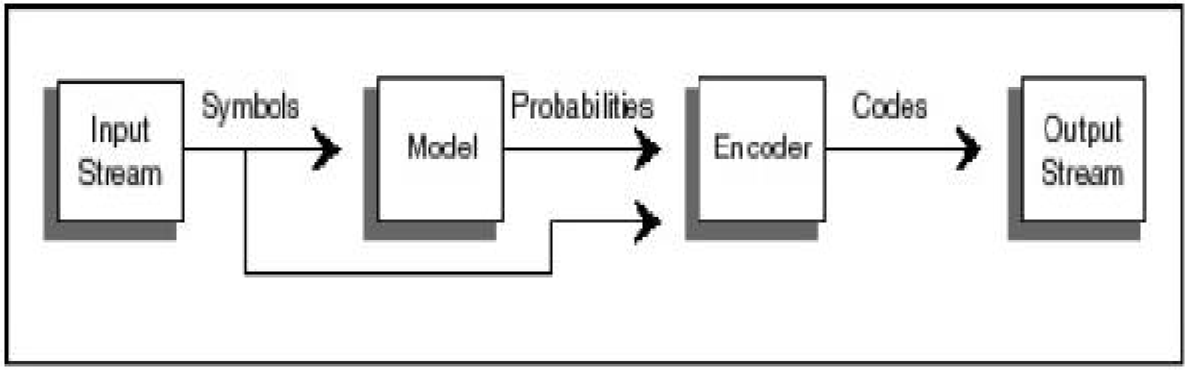
  – encode (compress) the input.
- Example 4.2 *Consider the Huffman encoding algorithm again. The coder assigns shorter codes to the more frequent symbols and longer codes to infrequent ones.*
- A similar structure applies to decoding algorithms.

Types of Model

- There are two types of compression algorithms, namely, *static, or adaptive compression, based on whether the model structure* may be updated during the process of compression or decompression
- Static (non-adaptive) system model remains unchanged during the compression or decompression process.
- Adaptive system model may be changed during the compression or decompression process according to the change of input (or feedback from the output).

Some adaptive algorithms actually build the model based on the input starting from an empty model.

Modelling + Coding



Data compression Encoder

Symmetric and Asymmetric Compression

- In some compression systems, the model for compression and that for decompression are identical.
- If they are identical, the compression system is called *symmetric, otherwise, it is said to*

be *non-symmetric.*

- *The compression using a symmetric system is called symmetric compression, and the compression using an asymmetric system is* called *asymmetric compression*

Coding methods

- In terms of the length of codewords used *before or after compression,* compression algorithms can be classified into the following categories:
  - Fixed-to-fixed: each symbol before compression is represented by a fixed number of bits (e.g. 8 bits in ASCII format) and is encoded as a sequence of bits of a fixed length after compression.
  - Example A:00, B:01, C:10, D:1

Coding Methods

- Fixed-to-variable: each symbol before compression is represented by a fixed number of bits and is encoded as a sequence of bits of different length.
- Example A:0; B:10; C:101; D:0101.