

# Lossless data compression

## Lecture 1

Lossless Data Compression, Fall 2005 - Mozilla Firefox

http://www.dani.au.dk/~brorille/DCS/lossless.html

### Lossless Data Compression, Fall 2005

Peter Bro Møhrsen

#### Virtual Handout

Peter Bro Møhrsen, *Course Notes for Data Compression*, 1  
John Kieffer, *Lectures on Source Coding*, Chapter 1, 2, 3.

#### Time and Place

Shannon-157, Mondays, 13:15-15 and Co-84-121, Thursdays, 14:15-16 during first quarter.

#### Structure of course

The first two weeks will be lectures, covering the basic theory. On September 12th, the java framework developed for the course will be presented. To pass the course, you must implement a variant of the compression scheme PFM using the framework and perform and report the results of certain experiments with your implementation. The final report must be handed in at the end of the quarter. Precise details will be given on September 12th. These projects may be done in groups with up to three students. *Please form such groups during the two first weeks of the quarter and tell me which groups have been formed on September 12th.* On October 24-26th, twenty-minute individual oral exams on the project and the course will be held. You'll get a grade on the 13-week, based on the project report and the exam.

#### Lectures

August 29th: Introduction to the course. The basic lossless source coding task. Prefix codes: Kraft-McMillan inequality. *Kieffer 1*, page 1-3, *Møhrsen 1*, page 1-5.  
<http://www.dani.au.dk/~brorille/DCS/notes5.1.ps>

## Data Compression

- **Lossless data compression:** Store/Transmit **big** files using few bytes so that the original files can be perfectly retrieved. Example: **zip**.
- **Lossy data compression:** Store/Transmit **big** files using few bytes so that the original files can be approximately retrieved. Example: **mp3**.
- **Motivation:** Save storage space and/or bandwidth.

comp.compression Frequently Asked Questions (part 1/3) - Mozilla Firefox

http://www.faqs.org/faqs/compression-faq/part1/

### 9.3.2 First details, by John Wallace <bwckey@ppf.trw.com>

I called WEB at (404) 514-8000 and they sent me some product literature as well as chatting for a few minutes with me on the phone. Their product is called DataFiles/16, and their claims for it are roughly those heard on the net.

According to their flyer:

"DataFiles/16 will compress all types of binary files to approximately one-sixteenth of their original size... regardless of the type of file (word processing document, spreadsheet file, image file, executable file, etc.), NO DATA WILL BE LOST by DataFiles/16." [Their capitalizations: 16:1 compression only promised for files >64K bytes in length.]

"Performed on a 386/25 machine, the program can complete a compression/decompression cycle on one megabyte of data in less than thirty seconds"

"The compressed output file created by DataFiles/16 can be used as the input file to subsequent executions of the program. This feature of the utility is known as recursive or iterative compression, and will enable you to compress your data files to a tiny fraction of the original size. In fact, virtually any amount of computer data can be compressed to under 1024 bytes using DataFiles/16 to compress its own output files multiple times. Then, by repeating in reverse the steps taken to perform the recursive compression, all original data can be decompressed to its original form without the loss of a single bit."

## Definition of Codec

- Let  $\Sigma$  be an alphabet and let  $S \subseteq \Sigma^*$  be a set of possible *messages*.
- A lossless **codec**  $(c,d)$  consists of
  - A **coder**  $c: S \rightarrow \{0,1\}^*$
  - A **decoder**  $d: \{0,1\}^* \rightarrow \Sigma^*$so that
  - $\forall x \in S: d(c(x))=x$

## Remarks

- It is necessary for  $c$  to be an injective map.
- If we do not worry about efficiency, we don't have to specify  $d$  if we have specified  $c$ .
- Terminology: Some times we just say "**code**" rather than "**codec**".
- Terminology: The set  $c(S)$  is called the set of **code words** of the codec. In examples to follow, we often just state the set of code words.

## Proposition

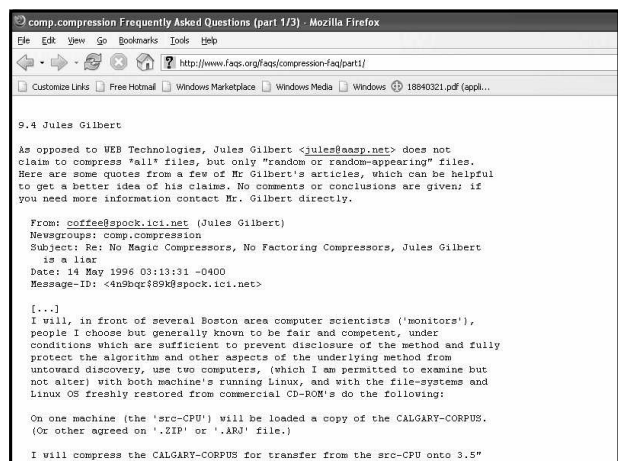
- Let  $S = \{0,1\}^n$ . Then, for any codec  $(c,d)$  there is some  $x \in S$ , so that  $|c(x)| \geq n$ .
- “*Compression is impossible*”

## Proposition

- For any message  $x$ , there is a codec  $(c,d)$  so that  $|c(x)|=1$ .
- “*The Encyclopedia Britannica can be compressed to 1 bit*”.

## Remarks

- We cannot compress *all* data. Thus, we must concentrate on compressing “relevant” data.
- It is trivial to compress data known in advance. We should concentrate on compressing data about which there is uncertainty.
- ***We will use probability theory as a tool to model uncertainty about relevant data.***



## Can random data be compressed?

- Suppose  $\Sigma = \{0,1\}$  and  $S = \{0,1\}^2$ .
- We know we cannot compress all data, but can we do well on the average?
- Let us assume the uniform distribution on  $S$  and look at the expected length of the code words.

!!!!!!!

Random data can be compressed well on the average!

..... ***There is something fishy going on***

## Definition of prefix codes

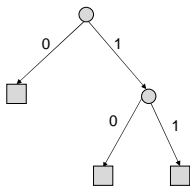
- A **prefix code**  $c$  is a code with the property that for all different messages  $x$  and  $y$ ,  $c(x)$  is not a prefix of  $c(y)$ .
- Example: **Fixed length** codes (such as ascii).
- Example:  $\{0,11,10\}$
- All codes in this course will be prefix codes.

## Proposition

- If  $c$  is a prefix code for  $S = \Sigma^1$  then  $c^n$  is a prefix code for  $S = \Sigma^n$  where
 
$$c^n(x_1 x_2 \dots x_n) = c(x_1) \cdot c(x_2) \dots c(x_n)$$

## Prefix codes and trees

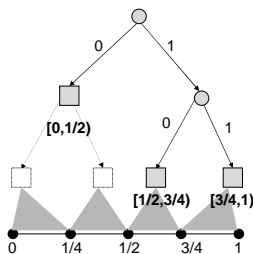
- Set of code words of a prefix code:  $\{0,11,10\}$ .



## Alternative view of prefix codes

- A prefix code is an assignment of the messages of  $S$  to the leaves of a rooted binary tree.
- The codeword of a message  $x$  is found by reading the labels on the edges on the path from the root of the tree to the leaf corresponding to  $x$ .

## Binary trees and the interval $[0,1)$



## Alternative view of prefix codes

- A prefix code is an assignment of the messages of  $S$  to disjoint **dyadic intervals**.
- A dyadic interval is a real interval of the form  $[k \cdot 2^{-m}, (k+1) \cdot 2^{-m})$  with  $k+1 \leq 2^m$ . The corresponding code word is the  $m$ -bit binary representation of  $k$ .

## Kraft-McMillan Inequality

- Let  $m_1, m_2, \dots$  be the lengths of the code words of a prefix code. Then,  $\sum 2^{-m_i} \leq 1$ .
- Let  $m_1, m_2, \dots$  be integers with  $\sum 2^{-m_i} \leq 1$ . Then there is prefix code  $c$  so that  $\{m_i\}$  are the lengths of the code words of  $c$ .

## Probability

- A **probability distribution**  $p$  on  $S$  is a map  $p: S \rightarrow [0,1]$  so that  $\sum_{x \in S} p(x) = 1$ .
- A  $U$ -valued **stochastic variable** is a map  $Y: S \rightarrow U$ .
- If  $Y: S \rightarrow \mathbf{R}$  is a stochastic variable, its **expected value**  $E[Y]$  is  $\sum_{x \in S} p(x) Y(x)$ .

## Self-entropy

- Given a probability distribution  $p$  on  $S$ , the **self-entropy** of  $x \in S$  is defined as
$$H(x) = -\log_2 p(x).$$
- The self-entropy of a message with probability 1 is 0 bits.
- The self-entropy of a message with probability 0 is  $+\infty$ .
- The self-entropy of a message with probability  $\frac{1}{2}$  is 1 bit.
- We often measure entropy in unit "bits"

## Entropy

- Given a probability distribution  $p$  on  $S$ , its **entropy**  $H[p]$  is defined as  $E[H]$ , i.e.
$$H[p] = -\sum_{x \in S} p(x) \log_2 p(x).$$
- For a stochastic variable  $X$ , its entropy  $H[X]$  is the entropy of its underlying distribution:
$$H[X] = -\sum_i \Pr[X=i] \log_2 \Pr[X=i]$$

## Facts

- The entropy of the uniform distribution on  $\{0,1\}^n$  is  $n$  bits. Any other distribution on  $\{0,1\}^n$  has strictly smaller entropy.
- If  $X_1$  and  $X_2$  are independent stochastic variables, then  $H(X_1, X_2) = H(X_1) + H(X_2)$ .
- For any function  $f$ ,  $H(f(X)) \leq H(X)$ .

## Shannon's theorem

- Let  $S$  be a set of messages and let  $X$  be an  $S$ -valued stochastic variable.
- For all prefix codes  $c$  on  $S$ ,
$$E[|c(X)|] \geq H[X].$$
- There is a prefix code  $c$  on  $S$  so that
$$E[|c(X)|] < H[X] + 1$$
In fact, for all  $x$  in  $S$ ,  $|c(x)| < H[x] + 1$ .