Technische Hochschule Rosenheim

Computer Science Fundamentals

Source Coding – Arithmetic Coding

Technische Hochschule Rosenheim
Winter 2021/22
Prof. Dr. Jochen Schmidt

Overview

- some notes on data compression
- arithmetic coding

Data Compression Methods

- Large variety of methods for data compression available

- Split into two groups
  - Lossless data compression
  - Lossy data compression

Technische Hochschule Rosenheim

Lossless data compression

- Objective of encoding
  - Redundancy reduction to zero if possible (Transfer time and storage space ↓)
- Essential requirement
  - The information contained in the data is retained without modification
  - This means that decoded data do not differ from the original data
- Application examples
  - Encoding of texts or tables

Technische Hochschule Rosenheim

Lossy data compression

- Objective of encoding
  - Reduction of amount of data beyond lossless data compression
- Information is essentially preserved, but some loss of information is accepted
  - This means that part of the information is lost
  - thus, the decoded data differ from the original data
- (substantially) higher compression rates can be achieved
- a lossless compression (like Huffman or arithmetic coding + run-length encoding) is typically part of the compression method
- Application examples
  - images, audio, or video files
    (perceptual psychological characteristics of the eyes/ears are taken into account)

Arithmetic Coding – Idea

- Principle:
  The entire message is assigned a floating-point number x in the interval $0 \leq x < 1$

- Single symbols can implicitly carry a non-integer information content
  - with Huffman, each symbol receives a code word with an integer length

- Arithmetic coding can usually reduce redundancy a little further
- Example

Source		Encoding
ESSEN	$\longrightarrow$	0.24704

Encoding		Decoding
0.24704	$\longrightarrow$	ESSEN

Arithmetic Coding – Principle

- Before the actual encoding of a message with n symbols, their relative frequencies are determined
- Starting from the interval [0,1[this is partitioned into n adjacent intervals
  - Each interval is assigned a symbol
  - The length of the intervals corresponds to the relative frequencies of the symbols
- the partitioning is applied recursively for each symbol of the message, resulting in nested partitionings

Arithmetic Coding – Example

- Message ESSEN is to be encoded arithmetically

- Necessary preparation steps
  - Determine relative frequencies (probabilities) p_i of the individual symbols
  - Assign an interval $[l(c), u(c)[$ to each symbol, where the length is proportional to the respective probability of occurrence

Symbol c	probability p_i	Interval $[l(c), u(c)[$
E	$^2/_5$	$[0.0, 0.4[$
S	$^2/_5$	$[0.4, 0.8[$
N	$^1/_5$	$[0.8, 1.0[$

Arithmetic Coding – Encoding

- Initialize lower and upper boundaries

$$L := 0.0$$
$$U := 1.0$$

- Read next input symbol c and calculate

$$s := U - L$$ … current length of the interval
$$U := L + s \cdot u(c)$$ … new upper bound, $u(c)$ from the table
$$L := L + s \cdot l(c)$$ … new lower bound, $l(c)$ from the table

until the end of the message is reached

- Result x (encoded input data)

$$x := \frac{L+U}{2}$$

(or $x := L$ or anything in the interval…)

Arithmetic Coding – Encoding Example

- Compression of the message ESSEN

Symbol c	Probability p_i	Interval $[l(c), u(c)[$
E	$^2/_5$	$[0.0, 0.4[$
S	$^2/_5$	$[0.4, 0.8[$
N	$^1/_5$	$[0.8, 1.0[$

c	s	U	L	
	-	1.0	0.0	… Initialization
E	1.0	0.4	0.0	
S	0.4	0.32	0.16	
S	0.16	0.288	0.224	
E	0.064	0.2496	0.224	
N	0.0256	0.2496	0.24448	

$$s := U - L$$
$$U := L + s \cdot u(c)$$
$$L := L + s \cdot l(c)$$

$$x := \frac{L + U}{2}$$

- The result is x = 0.24704

Arithmetic Coding – Decoding

Technische Hochschule Rosenheim

- Read code word x

- Repeat until all symbols are decoded (number of symbols must be known)

Output the symbol c that corresponds to the interval where x is located

$$s = u(c) - l(c)$$ … Length of the interval

$$x := \frac{x - l(c)}{s}$$ … New code word, where c has been removed

Arithmetic Coding – Decoding Example

Symbol c	Probability p_i	Interval $[l(c), u(c)[$
E	$^2/_5$	$[0.0, 0.4[$
S	$^2/_5$	$[0.4, 0.8[$
N	$^1/_5$	$[0.8, 1.0[$

We can gradually recover the original message from the encoded floating-point number x =0.24704)

x	c (output)	$u(c)$	$l(c)$	s
0.24704	E	0.4	0.0	0.4
0.6176	S	0.8	0.4	0.4
0.544	S	0.8	0.4	0.4
0.36	E	0.4	0.0	0.4
0.9	N	1.0	0.8	0.2

Output the symbol c that corresponds to the interval where x is located

$$s = u(c) - l(c)$$
$$x := \frac{x - l(c)}{s}$$

Arithmetic Coding – Exercise

- Encode the message IBIS arithmetically
  - sort the characters as they appear in the message for constructing the table (I, B, S)
  - this is arbitrary, but for reasons of comparison we'll all want to use the same order
- Decode the result

Encoding

$$s := U - L$$
$$U := L + s \cdot u(c)$$
$$L := L + s \cdot l(c)$$

$$x := \frac{L + U}{2}$$

Decoding

Output the symbol c that corresponds to the interval where x is located

$$s = u(c) - l(c)$$
$$x := \frac{x - l(c)}{s}$$

Arithmetic Coding – Issues

- The floating-point number must not be rounded during the coding process!
- Intervals are getting smaller and smaller with each symbol be encoded
  - However, CPUs have limited accuracy for floating-point numbers
  - From a certain limit, the “code number” can no longer be represented
  - But we require arbitrary (but finite) accuracy ⟶ needs to be implemented
- Probabilities of occurrence of the symbols must be known before encoding
  - Use pre-defined probabilities
  - Use of semi-adaptive/adaptive method of algorithm
- Much more computationally intensive than Huffman

Arithmetic Coding – Applications

- H.264/MPEG 4 AVC
  - (lossy) video encoding
  - z.B. Blu-ray or DVB-S2
  - Arithmetic encoding can optionally be used instead of Huffman for entropy encoding
- HEVC
  - also called: H.265/MPEG-H Part 2
  - Successor format of H.264
  - e.g., UHD-Bluy-ray (4k), DVB-T2, Streaming
  - arithmetic encoding mandatory, no Huffman