Entropy Coding

Notes

Chapter Contents

Notes

Huffman Coding

- Widely used data compression technique
- Can save 20% to 90% space
- Uses table of frequencies of occurence of characters to build optimal binary string for each character

Notes

Huffman Coding I

Example (from Cormen et al.)

- 100,000 character data file to be compressed
- Only 6 different characters in file with frequencies:

	a	b	c	d	e	f
frequency (1000×)	45	13	12	16	9	5
fixed length codeword	000	001	010	011	100	101
fixed codeword length	3	3	3	3	3	3
variable-length codeword	0	101	100	111	1101	1100
variable codeword length	1	3	3	3	4	4

- Fixed length codewords use:

$$3 \times 45 + 3 \times 13 + 3 \times 12 + 3 \times 16 + 3 \times 9 + 3 \times 5 = 300,000\text{bits}$$

- Variable length codewords use:

$$1 \times 45 + 3 \times 13 + 3 \times 12 + 3 \times 16 + 4 \times 9 + 4 \times 5 = 224,000\text{bits}$$

Notes

Huffman Coding II

25% saving!

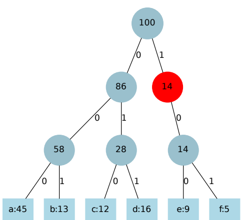
Example usage:

example string	abc	aabe	fadae
fixed length	000001010	000000001100	101000011000100
variable length	0101100	001011101	1100011101101

Notes

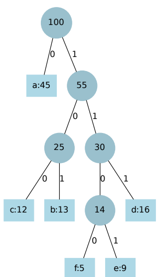
Trees for Coding Schemes

Fixed length coding:



Codes for each letter are read from the root of the tree down to the leaf node for the relevant character.

Huffman coding:



Notes

Trees for Coding Schemes I

- Trees very useful for coding and uncoding data

Steps to create a coding scheme:

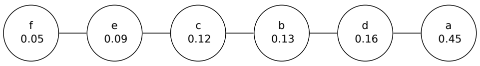
1. Calculate probabilities for each character by dividing the frequency (e.g. 45000 for a) by the total number of characters (here 100000)

	a	b	c	d	e	f	**total**
frequency	45000	13000	12000	16000	9000	5000	100000
probability	0.45	0.13	0.12	0.16	0.09	0.05	1

2. Sort characters in order of probability

f	e	c	b	d	a
0.05	0.09	0.12	0.13	0.16	0.45

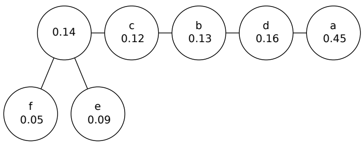
3. Link them as a graph



Notes

Trees for Coding Schemes II

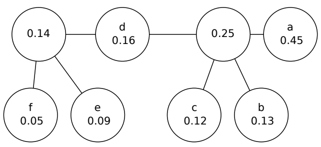
4. Merge the two nodes with the lowest probabilities



Notes

Trees for Coding Schemes III

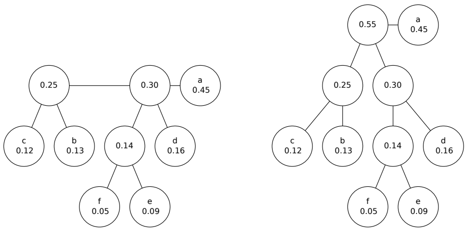
5. Sort the nodes in increasing order of probability



6. Repeat the above 2 steps for the nodes at the top level until a full binary tree is constructed...

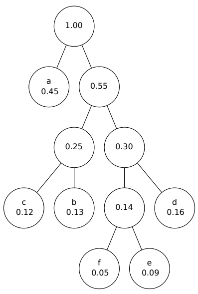
Notes

Trees for Coding Schemes IV



Notes

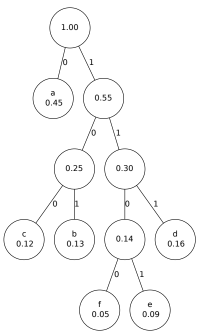
Trees for Coding Schemes V



Notes

Trees for Coding Schemes

7. Then assign 0 then 1 to each branch in the tree so that the two children of a parent are labelled 0 (on the left) and 1 (on the right)



- High probability letters have shorter codes
- Reduces overall length of the entire data
- Huffman codewords obtained by tracing each symbol from the root to the node for that symbol
- e.g. symbol f has codeword 1100, symbol d has codeword 111.

Notes

Problem with Huffman Coding

- Huffman coding is not always optimal
  - Optimal here means a communicated sequence is represented by using code words that are as short as possible.
- For optimal Entropy coding requires all symbols to have probabilities given by:

$$p(c) = \frac{1}{2^{l(c)}}$$

where $l(c)$ is the length of codeword c

- If the above is not true then the coding scheme is not optimal

Notes

Optimality of Huffman Coding

Average self information or Entropy for a sequence of symbols being communicated is given by:

$$H = -\sum p(c)\log_2(p(c)).$$

Average code word length:

$$L = \sum p(c)l(c).$$

Code words in Huffman coding represented in binary with 1 or more bits.

$$\therefore l(c) \geq 1$$

Thus

$$L \geq H.$$

Notes

Optimality of Huffman Coding

- A Hufman code for a sequence of symbols is optimal if the sequence is as short as possible.
- This means the coding is optimal when

$$L = H.$$

This occurs when length of code word $l(c)$ equal to self information:

$$l(c) = h(c)$$
$$l(c) = -\log_2(p(c)).$$

Thus, a coding is optimal when:

$$p(c) = \frac{1}{2^{l(c)}}.$$

Notes

Arithmetic Coding

Arithmetic coding is an alternative coding technique.
Summary of arithmetic coding steps:

1. Calculate cumulative probability ranges for each symbol
2. Select initial message interval with $[0, 1)$
3. Repeat following two steps for each symbol k in sequence:
   1. Divide current interval into subintervals, with sizes proportional to symbol's cumulative probabilities
   2. Select new subinterval as current interval

Notes

Arithmetic Coding

Arithmetic code following sequence:
cba
with probabilities:

	a	b	c	
probability, p_k	0.2	0.5	0.3	$\sum p_k = 1$

1. Calculate symbol ranges

a	$[0.00,$	$0.20)$
b	$[0.20,$	$0.70)$
c	$[0.70,$	$1.00)$

Notes

Arithmetic Coding

2. Determine message code

symbol ranges:

c	0.70	1.00
b	0.20	0.70
a	0.00	0.20

initial range	0.0	1.0
c	0.70	1.00
b	$(1.00 - 0.70) \times 0.20 + 0.70 = 0.76$	$(1.00 - 0.70) \times 0.70 + 0.70 = 0.91$
a	$(0.91 - 0.76) \times 0.00 + 0.76 = 0.76$	$(0.91 - 0.76) \times 0.20 + 0.76 = 0.79$

3. Thus, the message cba can be communicated with any number in the range:

$$[0.76, 0.79).$$

Notes

Arithmetic Coding

Arithmetic code following sequence:
`dabea`
with probabilities:

	a	b	c	d	e	f	
probability, p_k	0.45	0.13	0.12	0.16	0.09	0.05	$\sum p_k = 1$

1. Calculate symbol ranges

a	$[0.00,$	$0.45)$
b	$[0.45,$	$0.58)$
c	$[0.58,$	$0.70)$
d	$[0.70,$	$0.86)$
e	$[0.86,$	$0.95)$
f	$[0.95,$	$1.00)$

Notes

Arithmetic Coding

2. Determine message code

symbol ranges:

d	0.70	0.86
a	0.00	0.45
b	0.45	0.58
e	0.86	0.95
a	0.00	0.45

initial range	0.0	1.0
d	0.70	0.86
a	0.70	$(0.86-0.70)\times 0.45+0.70 = 0.772$
b	$(0.772 - 0.70) \times 0.45 + 0.70 = 0.7324$	$(0.772 - 0.70) \times 0.58 + 0.70 = 0.74176$
e	$(0.74176 - 0.7324) \times 0.86 + 0.7324 = 0.74045$	$(0.74176 - 0.7324) \times 0.95 + 0.7324 = 0.74129$
a	$(0.74129-0.74045)\times 0.00 + 0.74045 = 0.74045$	$(0.74129-0.74045)\times 0.45 + 0.74045 = 0.74082$

3. Thus, the message `dabea` can be communicated with any number in the range:

$$[0.74045, 0.74082).$$

Notes

Problem with Arithmetic Coding

- Arithmetic coding is not practical for realistic messages because it relies on repeated calculation of fractional amounts.
- These repeated calculations result in numbers that can not be stored or processed in unmodified form because computers are not able to represent floating point numbers to infinite precision.

Notes

Implementation of Arithmetic Coding

Arithmetic Coding can be made practical by

- Using integers to represent floating point numbers
  - Integers arithmetic is also faster.
- Integers have finite precision
  - Therefore shift left most digit out for communication, storage or further processing
  - Shift occurs when left most digit for lower and upper values become equal.

Notes

Arithmetic Coding Implementation

1. Determine message code cba with integer precision

symbol ranges:

c	0.70	1.00
b	0.20	0.70
a	0.00	0.20

1		2	3	4	5
c	L	0.70	7000		7000
	H	1.00	9999		9999
b	L	$(1.00-0.70)\times0.20+0.70=0.76$	7600		7600
	H	$(1.00-0.70)\times0.70+0.70=0.91$	9099		9099
a	L	$(0.91-0.76)\times0.00+0.76=0.76$	7600		7600
	H	$(0.91-0.76)\times0.20+0.76=0.79$	7899		7899

2. Thus, the message cba can be communicated with any number in the range:

$$[7600, 7900).$$

Notes

Arithmetic Coding Implementation

1. Determine message code dabea with integer precision *and shifting*

symbol ranges:

d	0.70	0.86
a	0.00	0.45
b	0.45	0.58
e	0.86	0.95
a	0.00	0.45

1		2	3	4	5
d	L	0.70	7000		7000
	H	0.86	8599		8599
a	L	0.70	7000	7	0000
	H	$(0.86-0.70)\times0.45+0.70=0.772$	7719	7	7199
b	L	$(0.72-0.0)\times0.45+0.0=0.324$	3240		3240
	H	$(0.72-0.0)\times0.58+0.0=0.4176$	4175		4175
e	L	$(0.4176-0.324)\times0.86+0.324=0.4045$	4045	4	0450
	H	$(0.4176-0.324)\times0.95+0.324=0.4129$	4128	4	1289
a	L	$(0.129-0.045)\times0.00+0.045=0.045$	0450	0	4500
	H	$(0.129-0.045)\times0.45+0.045=0.082$	0819	0	8199

2. Thus, the message dabea can be communicated with any number in the range:

$$[7404500, 7408200).$$

Notes

Decoding Integer Arithmetic Coding

- Opposite of encoding
- Start with first X digits of compressed stream (here $X = 4$ decimal digits as used in example).
- Initial range: $[0000, 9999]$
- Calculate index based on range and X digits to find next symbol by comparing with cumulative frequencies:
  `index=((code-L+1)×10-1)/(H-L+1)`
- Update range with
  `L=L+(H-L+1)×Lcumfreq[S]/10`
  `H=L+(H-L+1)×Hcumfreq[S]/(10-1)`
  where `Lcumfreq[S]` and `Hcumfreq[S]` are cumulative frequencies of symbol `S`.
- If left most digits of `L` and `H` are identical then shift left 1 position: `L` and `H`; update `code` with next value from compressed stream.

Notes

Information Content

Entropy of `dabea` sequence:

$$\lceil -\log_2(P_{\text{total}}) \rceil = \lceil -\log_2(0.16 \times 0.45 \times 0.13 \times 0.09 \times 0.45) \rceil = 12 \text{ bits}$$

Coded L value (74045):

$$0.1\ 0010\ 0001\ 0011\ 1101$$

Coded H value (74048):

$$0.1\ 0010\ 0001\ 0110\ 0010$$

Any number between these two:

$$0.\underbrace{1\ 0010\ 0001\ 01}_{\text{11 bits}}00\ 0000$$

Notes

Summary

- Entropy coding is a non-lossy compression algorithm used in image and video compression
- Huffman coding is a particular type of Entropy coding.
- Huffman coding generates variable length codewords for each symbol dependent on the probabilities for each symbol.
- The Huffman algorithm is only optimal for sets of symbols with probabilities equal to

$$p(c) = \frac{1}{2^{l(c)}}.$$

- Alternative Entropy coding techniques are needed to achieve higher compression ratios under more general conditions such as Arithmetic coding.

Notes