Lecture 15: Source Coding

Prof. Deniz Gunduz

Department of Electrical & Electronic Engineering
Imperial College London

Outline

- Average codeword length
- Fixed and variable length coding
- Source coding theorem
- Huffman coding
- Reference
  - ✓ [Haykin] Chapter 10

Source Entropy

- If symbol s_n has occurred, this corresponds to

$$I(s_n) = \log_2 \frac{1}{p_n} = -\log_2 p_n \text{ bits of information}$$

- For random variable S, expected value of $I(S)$ over the source alphabet

$$\mathbb{E}\{I(S)\} = \sum_{n=1}^{N} p_n I(s_n) = -\sum_{n=1}^{N} p_n \log_2 p_n$$

- Source entropy: average amount of information per source symbol

$$H(S) = -\sum_{n=1}^{N} p_n \log_2 p_n$$

- Units: bits/symbol

Example

Three-Symbol Alphabet

- A: occurs with probability 0.7
- B: occurs with probability 0.2
- C: occurs with probability 0.1

- Source entropy:

$$H(S) = -0.7\log_2(0.7) - 0.2\log_2(0.2) - 0.1\log_2(0.1) = 1.157 \text{ bits/symbol}$$

- How can we encode these symbols in order to transmit them?

- We need 2 bits/symbol if encoded as

$$A = 00, B = 01, C = 10 \quad \text{(fix-length coding)}$$

- Entropy prediction: the average amount of information is only 1.157 bits/symbol

- We are wasting bits!

Source Coding Theory

- Source encoding: concerned with minimizing the actual number of source bits that are transmitted to the user
- Channel encoding: concerned with introducing redundant bits to enable the receiver to detect and possibly correct errors that are introduced by the channel.
- What is the minimum number of bits required to transmit a particular symbol?
- How can we encode symbols so that we achieve (or at least come arbitrarily close to) this limit?

Example

Information Content of English

- Encoding English Words Letter-by-Letter
  - ✓ In English, on average there are 4.5 letters per word
  - ✓ With space (ignoring punctuation and capitalization) we need 5.5 characters per word
  - ✓ We need 5 bits to encode each letter (26 letters)
  - ✓ We need 27.5 bits per word
- Encoding English Words Word-by-Word
  - ✓ Assume 171,476 English words (from Google)
  - ✓ Equivalent to 18 bits per word ($2^{18} = 262,144$)
- Encoding English *Semantically*
  - ✓ . . . more efficient!

Frequencies of Words in English

1	the	26	they	51	when	76	come
2	be	27	we	52	make	77	its
3	to	28	say	53	can	78	over
4	of	29	her	54	like	79	think
5	and	30	she	55	time	80	also
6	a	31	or	56	no	81	back
7	in	32	an	57	just	82	after
8	that	33	will	58	him	83	use
9	have	34	my	59	know	84	two
10	I	35	one	60	take	85	how
11	it	36	all	61	people	86	our
12	for	37	would	62	into	87	work
13	not	38	there	63	year	88	first
14	on	39	their	64	your	89	well
15	with	40	what	65	good	90	way
16	he	41	so	66	some	91	even
17	as	42	up	67	could	92	new
18	you	43	out	68	them	93	want
19	do	44	if	69	see	94	because
20	at	45	about	70	other	95	any
21	this	46	who	71	than	96	these
22	but	47	get	72	then	97	give
23	his	48	which	73	now	98	day
24	by	49	go	74	look	99	most
25	from	50	me	75	only	100	us

Considering most frequent 8727 words ($\log_2(8727) = 14.4$ bits), the entropy of English word is found to be only 9.14 bits/word.

Can we reach this or beyond?

Average Codeword Length

- Definitions
  - ✓ l_n: number of bits used to code the n-th symbol
  - ✓ N: total number of symbols
  - ✓ p_n: probability of occurrence of symbol n

- Average codeword length

$$\bar{L} = \sum_{n=1}^{N} p_n l_n$$

- An idea to reduce average codeword length:
  - ✓ symbols that occur often should be encoded with short codewords
  - ✓ symbols that occur rarely may be encoded using long codewords

- Make sure that the codewords are uniquely decodable!

Codeword Length

- In a system with 2 symbols that are equally likely:
  - ✓ Probability of each symbol to occur: $p = 1/2$, $H(p) = 1$ bit
  - ✓ Best one can do: encode each with 1 bit only (0 or 1), $\bar{L} = 1 = H(p)$ bit
- In a system with 2 symbols that are unequally likely:
  - ✓ $H(p) < 1$ bit
  - ✓ Encode each with 1 bit only (0 or 1), $\bar{L} = 1 > H(p)$ bit
- A system with N ($N = 2^k$ for some integer k) symbols that are equally likely:
  - ✓ Probability of each symbol to occur: $p = 1/N$
  - ✓ One needs $\bar{L} = \log_2 N = k$ ($= -\log_2 p$) bits to represent the symbols
  - ✓ For example, $N = 4$ requires $L = 2$ bits
- What is the minimum average codeword length for a particular source?

Fixed Length Coding

Fixed length code: the same codeword length of different codewords.

Example: 4-symbol source $p(a_1) = 1/2$, $p(a_2) = 1/4$, $p(a_3) = p(a_4) = 1/8$

Symbol (codeword)	Prob	Code I	Code II	Code III
a_1	$1/2$	00	0	0
a_2	$1/4$	01	10	11
a_3	$1/8$	10	110	110
a_4	$1/8$	11	111	111

For example, using Code I:

$$a_1 a_3 a_4 a_3 \to 00, 10, 11, 10 \to 00101110$$
$$10110100 \to 10, 11, 01, 00 \to a_3 a_4 a_2 a_1$$

- Source entropy: $H(S) = \frac{1}{2}\log_2 2 + \frac{1}{4}\log_2 4 + 2 \times \frac{1}{8}\log_2 8 = 1.75$ bits
- Average length of codewords: $\bar{L} = 2 > H(S) = 1.75$

Fixed length coding is always **uniquely decodable** as long as you assign different symbols to different codewords!

Variable Length Coding

Variable length code: codewords may have different lengths

Example: 4-symbol source $p(a_1) = 1/2$, $p(a_2) = 1/4$, $p(a_3) = p(a_4) = 1/8$

Symbol (codeword)	Prob	Code I	Code II	Code III
a_1	$1/2$	00	0	0
a_2	$1/4$	01	10	11
a_3	$1/8$	10	110	110
a_4	$1/8$	11	111	111

Using Code II:

- Encoding: $a_1 a_3 a_4 a_1 \to 0, 110, 111, 0 \to 01101110$
- Decoding: $01101110 \to 0, 110, 111, 0 \to a_1 a_3 a_4 a_1$

Using code III:

- Encoding: $a_1 a_3 a_4 a_1 \to 0, 110, 111, 0 \to 01101110$
- Decoding:
  - ✓ $01101110 \to 0, 110, 111, 0 \to a_1 a_3 a_4 a_1$
  - ✓ $01101110 \to 0, 11, 0, 111, 0 \to a_1 a_2 a_1 a_4 a_1$

Not uniquely decodable!

Some variable length codes are not uniquely decodable, and we only consider **uniquely decodable** coding subsequently.

Source Coding Theorem

Source Coding Theorem

Given a discrete memoryless source of entropy $H(S)$, average codeword length $\bar{L}$ for any **uniquely decodable source coding scheme** is (lower) bounded by $H(S)$, that is,

$$\bar{L} \geq H(S)$$

- Basic Idea: choosing codeword lengths so that more-probable sequences have shorter codewords
- Code Construction:
  - ✓ Sort source symbols in order of decreasing probability
  - ✓ Take two smallest $p(x_i)$ and assign each a different bit (i.e., 0 or 1), then merge into a single symbol
  - ✓ Repeat until only one symbol remains
- Properties:
  - ✓ Huffman Coding (among other algorithms): uniquely decodable with average coding length satisfying $H(S) \leq \bar{L} < H(S) + 1$
  - ✓ The shortest average codeword length
  - ✓ Easy to implement this algorithm: used in JPEG, MP3, ...

Example

Huffman coding

Compound Symbol using Huffman Coding

- Two symbol source: two symbols s_1, s_2
  - ✓ probabilities $\Pr(s_1) = p_1$, $\Pr(s_2) = p_2$
  - ✓ $H_1(S) = -p_1 \log_2 p_1 - p_2 \log_2 p_2$
  - ✓ Average length of Huffman code: $H_1(S) \le \bar{L}_1 < H_1(S) + 1$

- Compound-symbol source by combining every two symbols:
  - ✓ Four compound symbols $s_1 s_1$, $s_1 s_2$, $s_2 s_1$, $s_2 s_2$
  - ✓ Probabilities

$$\Pr(s_1 s_1) = p_1^2, \Pr(s_1 s_2) = \Pr(s_2 s_1) = p_1 p_2, \Pr(s_2 s_2) = p_2^2$$

  - ✓ Compound-symbol source entropy

$$\begin{aligned} H_2(S) &= -\sum_{i,j} p_i p_j \log(p_i p_j) = -\sum_{i,j} p_i p_j \log(p_i) - \sum_{i,j} p_i p_j \log(p_j) \\ &= -\sum_i p_i \log(p_i) - \sum_j p_j \log(p_j) \\ &= 2H_1(S) \end{aligned}$$

  - ✓ Average length of Huffman code per the compound-symbol: $\bar{L}_2$

$$H_2(S) \le \bar{L}_2 < H_2(S) + 1$$

  - ✓ Average length per symbol: $\bar{L}_2/2$

$$2H_1(S) \le \bar{L}_2 < 2H_1(S) + 1 \Rightarrow H_1(S) \le \frac{\bar{L}_2}{2} < H_1(S) + \frac{1}{2}$$

Compound Symbol using Huffman Coding

- Compound-symbol source by combining K symbols:
  - ✓ Probability $\Pr(s_{n_1} s_{n_2} \dots s_{n_K}) = p_{n_1} p_{n_2} \dots p_{n_K} = \prod_{k=1}^{K} p_{n_k}$
  - ✓ Compound-symbol source entropy

$$H_K(S) = K H_1(S)$$

  - ✓ Average length of Huffman code per the compound-symbol: $\bar{L}_K$

$$H_K(S) \leq \bar{L}_K < H_K(S) + 1$$

  - ✓ Average length per symbol: $\bar{L}_K / K$

$$H_1(S) \leq \frac{\bar{L}_K}{K} < H_1(S) + \frac{1}{K}$$

- When $K \to \infty$,

$$H_1(S) \leq \lim_{K \to \infty} \frac{\bar{L}_K}{K} \leq H_1(S) + \lim_{K \to \infty} \frac{1}{K}$$

Average length per symbol:

$$\lim_{K \to \infty} \frac{\bar{L}_K}{K} = H_1(S)$$

- A drawback of Huffman coding: requiring knowledge of a probabilistic model, which is not always available a priori.
- Lempel-Ziv coding overcomes this practical limitation and has become the standard algorithm for file compression.
  - ✓ In principle it 'learns' the distribution of a file in an online fashion
  - ✓ compress, gzip, GIF, TIFF, PDF, modem, ...
  - ✓ A text file can typically be compressed to half of its original size.

Note