Information Theory

Chapter III: Source coding

What does coding do?

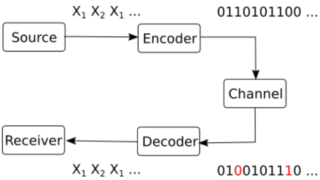

Figure 1: Communication system

- Why coding?

1. Source coding
    - Convert source messages to channel symbols (for example 0,1)
    - Minimize number of symbols needed
    - Adapt probabilities of symbols to maximize mutual information
2. Error control
    - Protection against channel errors / Adds new (redundant) symbols

Source-channel separation theorem

Source-channel separation theorem (informal):

- It is possible to obtain the best reliable communication by performing the two tasks separately:
  1. Source coding: to minimize number of symbols needed
  2. Error control coding (channel coding): to provide protection against noise

Source coding

- Assume we code for transmission over ideal channels with no noise
- Transmitted symbols are perfectly recovered at the receiver
- Main concerns:
  - minimize the number of symbols needed to represent the messages
  - make sure we can decode the messages
- Advantages:
  - Efficiency
  - Short communication times
  - Can decode easily

Definitions

- Let $S = \{s_1, s_2, ...s_N\}$ = an input discrete memoryless source
- Let $X = \{x_1, x_2, ...x_M\}$ = the alphabet of the code
  - Example: binary: $\{0,1\}$
- A **code** is a mapping from S to the set of all codewords:

$$C = \{c_1, c_2, ...c_N\}$$

Message	Codeword
s_1	$c_1 = x_1 x_2 x_1 ...$
s_2	$c_2 = x_1 x_2 x_2 ...$
...	....
s_N	$c_3 = x_2 x_2 x_2 ...$

- Decoding: given a sequence of symbols, deduce the original sequence of messages
- Codeword length l_i = the number of symbols in c_i

Example: ASCII code

Letter	ASCII Code	Binary	Letter	ASCII Code	Binary
a	097	01100001	A	065	01000001
b	098	01100010	B	066	01000010
c	099	01100011	C	067	01000011
d	100	01100100	D	068	01000100
e	101	01100101	E	069	01000101
f	102	01100110	F	070	01000110
g	103	01100111	G	071	01000111
h	104	01101000	H	072	01001000
i	105	01101001	I	073	01001001
j	106	01101010	J	074	01001010
k	107	01101011	K	075	01001011
l	108	01101100	L	076	01001100
m	109	01101101	M	077	01001101
n	110	01101110	N	078	01001110
o	111	01101111	O	079	01001111
p	112	01110000	P	080	01010000
q	113	01110001	Q	081	01010001
r	114	01110010	R	082	01010010
s	115	01110011	S	083	01010011
t	116	01110100	T	084	01010100
u	117	01110101	U	085	01010101
v	118	01110110	V	086	01010110
w	119	01110111	W	087	01010111
x	120	01111000	X	088	01011000
y	121	01111001	Y	089	01011001
z	122	01111010	Z	090	01011010

Figure 2: ASCII code (partial)

Average code length

- How to measure representation efficiency of a code?
- **Average code length** = average of the codeword lengths:

$$\bar{l} = \sum_i p(s_i) l_i$$

- The probability of a codeword = the probability of the corresponding message
- Smaller average length: code more efficient (better)
- How small can the average length be?

Definitions

A code can be:

- **non-singular**: all codewords are different
- **uniquely decodable**: for any received sequence of symbols, there is only one corresponding sequence of messages
  - i.e. no sequence of messages produces the same sequence of symbols
  - i.e. there is never a confusion at decoding
- **instantaneous** (also known as **prefix-free**): no codeword is prefix to another code
  - A *prefix* = a codeword which is the beginning of another codeword

Examples: at the blackboard

The graph of a code

Example at blackboard

Instantaneous codes are uniquely decodable

Theorem

- An instantaneous code is uniquely decodable
- (The converse is not necessary true; there exist uniquely decodable codes which are not instantaneous)

Proof

- blackboard

Comments:

- How to decode an instantaneous code: graph-based decoding
- Advantage on instantaneous code over uniquely decodable: simple decoding

Existence of instantaneous codes

- When can there an instantaneous code exist?

Kraft inequality theorem:

- There exists an instantaneous code with D symbols and codeword lengths $l_1, l_2, \ldots l_n$ if and only if the lengths satisfy the following inequality:

$$\sum_i D^{-l_i} \leq 1.$$

Proof:

- At blackboard

Comments:

- If lengths do not satisfy this, no instantaneous code exists
- If the lengths of a code satisfy this, that code can be instantaneous or not (there exists an instantaneous code, but not necessarily that one)
- Kraft inequality means that the codewords lengths cannot be all very small

Instantaneous codes with equality in Kraft

- From the proof $=>$ we have equality in the relation

$$\sum_i D^{-l_i} = 1$$

  only if the lowest level is fully covered $<=>$ no unused branches
- For an instantaneous code which satisfies Kraft with equality, all the graph branches terminate with codewords (there are no unused branches)

Kraft inequality for uniquely decodable codes

- Instantaneous codes must obey Kraft inequality
- How about uniquely decodable codes?

McMillan theorem:

- An uniquely decodable code satisfies the Kraft inequality:

$$\sum_i D^{-l_i} \leq 1.$$

Consequence:

- For every uniquely decodable code, there exists in instantaneous code with the same lengths.
- Even though the class of uniquely decodable codes is larger than that of instantaneous codes, we have no benefit.
- We can always use just instantaneous codes.

Finding an instantaneous code for given lengths

- How to find an instantaneous code with code lengths $\{l_i\}$

1. Check that lengths satisfy Kraft relation
2. Draw graph
3. Assign nodes in a certain order (e.g. descending probability)

- Easy, standard procedure
- Example: blackboard

Optimal codes

- We want to minimize the **average length** of a code:

$$\bar{l} = \sum_i p(s_i) l_i$$

- But the lengths must obey the Kraft inequality (for uniquely decodable)

$$\textbf{minimize} \ \sum_i p(s_i) l_i$$
$$\text{subject to} \ \sum_i D^{-l_i} \leq 1$$

- The optimal values are:

$$\boxed{l_i = -\log(p(s_i))}$$

- Rigorous proof: at blackboard (method of Lagrange multiplier)
- Intuition: using $l_i = -\log(p(s_i))$ satisfies Kraft with equality, so the lengths cannot be any shorter, in general

Entropy = minimal codeword lengths

- If the optimal values are:

$$l_i = -\log(p(s_i))$$

- Then the minimal average length is:

$$\min \bar{l} = \sum_i p(s_i) l_i = -\sum_i p(s_i) \log(p(s_i)) = H(S)$$

Average length >= entropy

The average length of an uniquely decodable code cannot be smaller than the source entropy

$$H(S) \leq \bar{l}$$

Non-optimal codes

- Problem: $-\log(p(s_i))$ might not be an integer number
- $l_i = -\log(p(s_i))$ only when probabilities are power of 2 (*dyadic distribution*)
- Shannon's solution: round to bigger integer

$$l_i = \lceil -\log(p(s_i)) \rceil$$

Shannon coding

- Shannon coding:
  1. Arrange probabilities in descending order
  2. Use codeword lengths $l_i = \lceil -\log(p(s_i)) \rceil$
  3. Find an instantaneous code for these lengths
- Simple scheme, better algorithms are available
  - Example: compute lengths for $S : (0.9, 0.1)$
- But still enough to prove fundamental results

Average length of Shannon code

- The average length of a Shannon code satisfies

$$H(S) \leq \bar{l} < H(S) + 1$$

- Proof:

1. The first inequality is because H(S) is minimum length
2. The second inequality:
   2.1 Use Shannon code:

   $$l_i = \lceil -\log(p(s_i)) \rceil = -\log(p(s_i)) + \epsilon_i$$

   where $0 \leq \epsilon_i < 1$
   2.2 Compute average length:

   $$\bar{l} = \sum_i p(s_i) l_i = H(S) + \sum_i p(s_i)\epsilon_i$$

   2.3 Since $\epsilon_i < 1 => \sum_i p(s_i)\epsilon_i < \sum_i p(s_i) = 1$

Average length of Shannon code

- Shannon code approaches minimum possible lengths up to at most 1 extra bit
  - That's not bad at all
  - There exist even better codes, in general
- Can we get even closer to the minimum length?
- Yes, as close as we want! See next slide.

Shannon's first theorem

Shannon's first theorem (coding theorem for noiseless channels):

- One can always compress messages from a source S with an average length as close as desired to H(S), but never below H(S) (for infinitely long sequences of messages)

Proof:

- Average length can never go below H(S) because this is minimum
- How can it get very close to H(S) (from above)?
  1. Use n-th order extension S^n of S
  2. Use Shannon coding for S^n, so it satisfies

$$H(S^n) \leq \overline{l_{S^n}} < H(S^n) + 1$$

  3. But $H(S^n) = nH(S)$, and **average length per message of S is**

$$\overline{l_S} = \frac{\overline{l_{S^n}}}{n}$$

because messages of S^n are just n messages of S glued together

Shannon's first theorem

- Continuing:
  4. So, dividing by n:
  $$\boxed{H(S) \leq \overline{l_S} < H(S) + \frac{1}{n}}$$
  5. If extension order $n \to \infty$, then
  $$\overline{l_S} \to H(S)$$

Comments:

- Shannon's first theorem says what entropy H(S) means:
- The entropy H(S) means the minimum number of bits required to describe a message from S, in general
- For any distribution we can approach H(S) to any desired accuracy using extensions of large order
  - The complexity is too large for large n, so in practice we settle with a close enough value
- Other codes are even better the Shannon coding

Efficiency and redundancy of a code

- **Efficiency** of a code (M = size of code alphabet):

$$\eta = \frac{H(S)}{\bar{l} \log M}$$

- **Redundancy** of a code:

$$\rho = 1 - \eta$$

- These measures indicate how close is the average length to the optimal value
- When $\eta = 1$: **optimal code**
  - for example when $l_i = -\log(p(s_i))$

Coding with the wrong code

- Consider a source with probabilities $p(s_i)$
- We use a code designed for a different source: $l_i = -\log(q(s_i))$
- The message probabilities are $p(s_i)$ but the code is designed for $q(s_i)$
- How much do we lose?
- Example: different languages

- Codeword lengths are not optimal for this source => increased $\bar{l}$
- If code were optimal, best average length = entropy $H(S)$:

$$\overline{l_{optimal}} = -\sum p(s_i) \log p(s_i)$$

- The actual average length:

$$\overline{l_{actual}} = \sum p(s_i) l_i = -\sum p(s_i) \log q(s_i)$$

The Kullback–Leibler distance

- Difference is:

$$\overline{l_{actual}} - \overline{l_{optimal}} = \sum_i p(s_i) \log(\frac{p(s_i)}{q(s_i)}) = D_{KL}(p, q)$$

Definition: the Kullback–Leibler distance of two distributions is

$$D_{KL}(p, q) = \sum_i p(i) \log(\frac{p(i)}{q(i)})$$

Properties:

- Always positive:

$$D_{KL}(p, q) \geq 0, \forall p, q$$

- Equals 0 only when the two distributions are identical

$$D_{KL}(p, q) = 0 <=> p(s_i) = q(s_i), \forall i$$

The Kullback–Leibler distance

Where is the Kullback–Leibler distance used:

- Using a code for a different distribution:
  - Average length is increased with $D_{KL}(p, q)$
- Definition of mutual information:
  - Distance between $p(x_i \cap y_j)$ and the distribution of two independent variables $p(x_i) \cdot p(y_j)$

$$I(X, Y) = \sum_{i,j} p(x_i \cap y_j) \log(\frac{p(x_i \cap y_j)}{p(x_i)p(y_j)})$$

Shannon-Fano coding (binary)

Shannon-Fano (binary) coding procedure:

1. Sort the message probabilities in descending order
2. Split into two subgroups as nearly equal as possible
3. Assign first bit 0 to first group, first bit 1 to second group
4. Repeat on each subgroup
5. When reaching one single message $=>$ that is the codeword

Example: blackboard

Comments:

- Shannon-Fano coding does not always produce the shortest code lengths
- Connection: yes-no answers (see example from source chapter)

Huffman coding (binary)

Huffman coding procedure (binary):

1. Sort the message probabilities in descending order
2. Join the last two probabilities, insert result into existing list, preserve descending order
3. Repeat until only two messages are remaining
4. Assign first bit 0 and 1 to the final two messages
5. Go back step by step: every time we had a sum, append 0 and 1 to the end of existing codeword

Example: blackboard

Properties of Huffman coding

Properties of Huffman coding:

- Produces a code with the **smallest average length** (better than Shannon-Fano)
- Assigning 0 and 1 can be done in any order $=>$ different codes, same lengths
- When inserting a sum into existing list, may be equal to another value $=>$ options
  - we can insert above, below or in-between equal values
  - leads to codes with different *individual* lengths, but same *average* length
- Some better algorithms exist which do not assign a codeword to every single message (they code a while sequence at once, not every message)

Huffman coding (M symbols)

General Huffman coding procedure for codes with M symbols:

- Have M symbols $\{x_1, x_2, ...x_M\}$
- Add together the last M symbols
- When assigning symbols, assign all M symbols
- **Important**: at the final step must have M remaining values
  - May be necessary to add *virtual* messages with probability 0 at the end of the initial list, to end up with exactly M messages in the end
- Example : blackboard

Comparison of Huffman and Shannon-Fano coding

Comparison of binary Huffman and Shannon-Fano example:

$$p(s_i) = \{0.35, 0.17, 0.17, 0.16, 0.15\}$$

Coding followed by channel

- For every symbol $x_i, i \in \{1, 2 \ldots M\}$ we can compute the average number of symbols x_i in a codeword

$$\overline{l_{x_i}} = \sum_i p(s_i) l_{x_i}(s_i)$$

- (here $l_{x_i}(s_i) =$ number of symbols x_i in codeword of s_i)
- Divide by average length $=>$ obtain probability (frequency) of symbol x_i

$$p(x_i) = \frac{\overline{l_{x_i}}}{\overline{l}}$$

- These are the symbol probabilities at the input of the following channel
- Example: binary code ($\overline{l_0}$, $\overline{l_1}$, $p(0)$, $p(1)$)

Chapter summary

- Average length: $\bar{l} = \sum_i p(s_i) l_i$
- Code types: instantaneous $\subset$ uniquely decodable $\subset$ non-singular
- All instantaneous or uniqualy decodable code must obey Kraft inequality

$$\sum_i D^{-l_i} \leq 1$$

- Optimal codes: $l_i = -\log(p(s_i))$, $\overline{l_{min}} = H(S)$
- Shannon's first theorem: use n-th order extension of S, S^n:

$$\boxed{H(S) \leq \overline{l_S} < H(S) + \frac{1}{n}}$$

  - average length can get as close as possible to $H(S)$
  - average length can never be smaller than $H(S)$
- Coding techniques:
  - Shannon: ceil optimal codeword lengths (round to upper)
  - Shannon-Fano: split in two groups approx. equal
  - Huffman: best