Fundamentals of Mathematical Informatics

Existence of Optimal Source Codes

Francesco Buscemi

Lecture Three

Reminder from lecture two: Kraft's inequality and the noiseless source coding theorem

Let $\mathscr{S}$ be an i.i.d. information source with word set $\mathcal{W} = \{w_1, \cdots, w_N\}$, probability distribution $(p_1, \cdots, p_N)$, and entropy rate $H(\mathscr{S}) \stackrel{\text{def}}{=} H(p_1, \cdots, p_N)$. Let Σ be a D-ary alphabet $\{\sigma_1, \cdots, \sigma_D\}$.

Kraft's Inequality

There exists a prefix code $f : \mathcal{W} \to \Sigma^*$ with word lengths $l_1, l_2, \cdots, l_N$ iff $\sum_{i=1}^N D^{-l_i} \leqslant 1$.

Noiseless source-coding theorem

Any D-ary prefix code $f : \mathcal{W} \to \Sigma^*$ must satisfy the following inequality:

$$\langle f \rangle \stackrel{\text{def}}{=} \sum_{i=1}^N p_i l_i \geqslant \frac{H(\mathscr{S})}{\log_2 D}.$$

Moreover, there always exists a D-ary prefix code $\bar{f} : \mathcal{W} \to \Sigma^*$ such that

$$\langle \bar{f} \rangle < \frac{H(\mathscr{S})}{\log_2 D} + 1.$$

Reminder from lecture two: proof of direct part of noiseless source-coding theorem

- Imagine that, for each p_i, there exists an integer $\bar{l}_i$ such that $D^{-\bar{l}_i} = p_i$, i.e., $\bar{l}_i = -\frac{\log_2 p_i}{\log_2 D}$.
- If that is true, then we know that there exists (and we know how to construct) a D-ary prefix code $\bar{f}$ with word lengths $\bar{l}_i$, because Kraft's inequality is automatically satisfied: $\sum_i D^{-\bar{l}_i} = \sum_i p_i = 1$.
- The average length of such a code is $\langle \bar{f} \rangle = \sum_i p_i \bar{l}_i = \sum_i p_i (-\frac{\log_2 p_i}{\log_2 D}) = \frac{H(\mathscr{S})}{\log_2 D}$, which is already optimal.
- **Problem:** the lengths $\bar{l}_i$ are not, in general, *integer numbers*!
- To avoid such a problem, choose $l_i^* = \lceil \bar{l}_i \rceil$ for all i. (The symbol $\lceil x \rceil$ denotes the 'ceiling' of x, i.e., the smallest integer greater than or equal to x.)
- This implies that $\bar{l}_i \leqslant l_i^* < \bar{l}_i + 1$ for all i.
- Again, Kraft's inequality is obeyed since $\sum_i D^{-l_i^*} \leqslant \sum_i D^{-\bar{l}_i} = 1$.
- Only the average length is worse, because $\sum_i p_i l_i^* \geqslant \sum_i p_i \bar{l}_i$, but not too much, because

$$\sum_i p_i l_i^* < \sum_i p_i (\bar{l}_i + 1) = \frac{H(\mathscr{S})}{\log_2 D} + \sum_i p_i = \frac{H(\mathscr{S})}{\log_2 D} + 1. \ \square$$

Shannon codes

- From the proof, we can get a 'good' code if we choose the word lengths such that $l_i = \lceil -\log_2 p_i \rceil = \left\lceil \log_2 \frac{1}{p_i} \right\rceil$. (In this lecture we will mostly consider binary codes.)
- Such codes are called **Shannon codes**.
- Shannon codes, even being 'good' *in average*, can be quite bad for single codewords.
- **Example.** Let $\mathcal{W} = \{w_1, w_2\}$ with $p_1 = \frac{127}{128}$ and $p_2 = \frac{1}{128} = 2^{-7}$. Then, $l_1 = \lceil 0.003 \rceil = 1$ and $l_2 = 7$. However, we can perfectly encode both w_1 and w_2 using just one bit.
- In this lecture we will study an optimal construction, called **Huffman coding**, that circumvents this problem.
- **Remark.** Shannon codes and Huffman codes are variable-length codes (i.e., the word lengths l_i vary with i).

Huffman codes: first example

- Consider a source with

w_1	w_2	w_3	w_4	w_5
0.25	0.25	0.2	0.15	0.15

and entropy $H(\mathscr{S}) \approx 2.285$

- Keep grouping the two least likely words, until only two words are left.

Start with w_4 and w_5

w_1	w_2	w_3	$w_4 \& w_5$
0.25	0.25	0.2	0.3

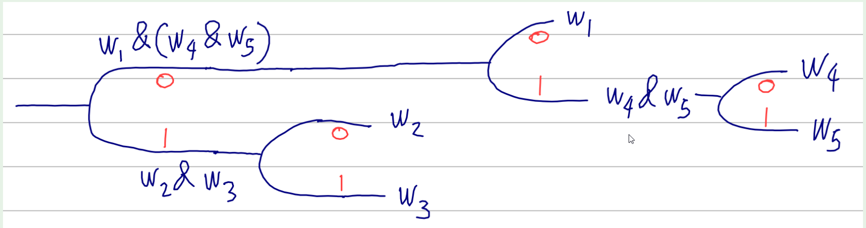
Then group w_2 and w_3

w_1	$w_2 \& w_3$	$w_4 \& w_5$
0.25	0.45	0.3

Then group w_1 and $w_4 \& w_5$

$w_1 \& (w_4 \& w_5)$	$w_2 \& w_3$
0.55	0.45

- Assign codewords working backward, as in the figure:



- The code is

w_1	w_2	w_3	w_4	w_5
00	10	11	010	011

- The average length is $2(0.25) + 2(0.25) + 2(0.2) + 3(0.15) + 3(0.15) = 2.3$.

Huffman codes: second example

- The source is

w_1	w_2	w_3	w_4
$1/3$	$1/3$	$1/4$	$1/12$

with entropy $H(\mathscr{S}) \approx 1.855$

- First solution:

w_1	w_2	$w_3 \& w_4$
$1/3$	$1/3$	$1/3$

$\rightarrow$

$w_1 \& w_2$	$w_3 \& w_4$
$2/3$	$1/3$

$\rightarrow$

w_1	w_2	w_3	w_4
00	01	10	11

- Average length is 2.
- Second solution:

w_1	w_2	$w_3 \& w_4$
$1/3$	$1/3$	$1/3$

$\rightarrow$

w_1	$w_2 \& (w_3 \& w_4)$
$1/3$	$2/3$

$\rightarrow$

w_1	w_2	w_3	w_4
0	10	110	111

- Average length is: $1(1/3) + 2(1/3) + 3(1/4) + 3(1/12) = 2$.
- Huffman coding is not always unique. **Question:** is the average length always the same in such cases? Why? **Answer:** yes, it must be the same, because Huffman coding is optimal! (We will see this in a minute.)

Again: Shannon codes *versus* Huffman codes

- Take again the previous example:

w_1	w_2	w_3	w_4
$1/3$	$1/3$	$1/4$	$1/12$

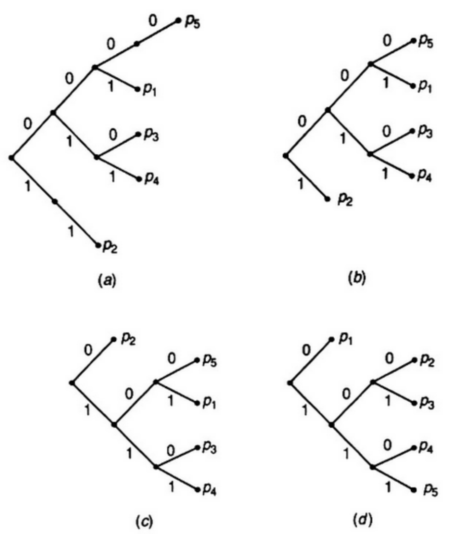
- The second Huffman code we constructed was

w_1	w_2	w_3	w_4
0	10	110	111

- Remember the Shannon coding technique: it assigns codewords of length $l_i = \left\lceil \log_2 \frac{1}{p_i} \right\rceil$.
- Look at the source word w_3: Huffman coding assigns a codeword of length 3, Shannon coding assigns a codeword of length 2.
- But the average length for Huffman is 2, while for Shannon is $\frac{2}{3}\lceil \log_2 3 \rceil + \frac{1}{4}\lceil \log_2 4 \rceil + \frac{1}{12}\lceil \log_2 12 \rceil = \frac{2}{3}(2) + \frac{1}{4}(2) + \frac{1}{12}(4) = 13/6 > 2$.
- While for single codewords either Huffman or Shannon can be shorter, **Hufmann coding is always shorter on average**. (Actually it is always the *shortest*, because Huffman coding is optimal—as we are going to see next.)
- **Conclusion.** We are looking at the average rate, not at the single word lengths.

Optimality of Huffman coding (proof idea)

Taken from: Cover & Thomas, *Elements of Information Theory (Second Ed.)*, p.124.



- Take a source $\mathscr{S}$ of five words with probabilities $p_1 \geqslant p_2 \geqslant \cdots \geqslant p_5$.
- Consider some prefix code for $\mathscr{S}$, like the one in (a).
- We can 'prune' branches without siblings and get (b).
- We then order codewords so that shorter codewords are on top, longer are at the bottom of the tree: we get (c).
- We finally reorder codewords from top to bottom, according to their probabilities: we get (d).
- What are the properties of the code in (d)?
  1. All branches have a sibling.
  2. If $p_i > p_j$ then $l_i \leqslant l_j$.
- **Recursive/iterative consistency:** grouping together two least likely words, (1) and (2) still hold.
- Codes satisfying the above conditions are optimal.
- Huffman codes are, by construction, optimal.
- **Remark.** There are many optimal codes: for example, inverting bits ($0 \leftrightarrow 1$) of an optimal code gives another optimal code.

D-ary Huffman codes

- They are like binary Huffman codes, but we group the D (instead of two) least likely source words at each step.
- So, each step has $D-1$ words less than the previous one.
- The last step has exactly D words.
- Therefore: the initial number of source words N must be such that $N = D + k(D-1)$, for some $k \in \mathbb{N}$. Equivalently, $(N-1)$ must be an integer multiple of $(D-1)$.
- **Question:** what happens otherwise? **Answer:** otherwise we append extra 'dummy' source words each having zero probability of occurrence.

D-ary Huffman codes: an example

- Find a ternary ($D = 3$) Huffman code for

w_1	w_2	w_3	w_4	w_5	w_6
0.25	0.25	0.2	0.1	0.1	0.1

- $N = 6$, therefore $(N-1) \neq k(D-1)$.
- We append an extra dummy word, ☺, so the source becomes

w_1	w_2	w_3	w_4	w_5	w_6	☺
0.25	0.25	0.2	0.1	0.1	0.1	0.0

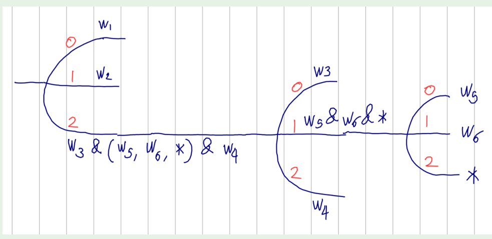
- We then proceed

w_1	w_2	w_3	w_4	$(w_5 \& w_6 \& ☺)$
0.25	0.25	0.2	0.1	0.2

-

w_1	w_2	$(w_3 \& (w_5 \& w_6 \& ☺) \& w_4)$
0.25	0.25	0.5

- We obtain



that is

w_1	w_2	w_3	w_4	w_5	w_6	☺
0	1	20	22	210	211	212

- Average length is $0.25(1) + 0.25(1) + 0.2(2) + 0.1(2) + 0.1(3) + 0.1(3) + 0.0(3) = 1.7$.
- The entropy bound is $\frac{H(\mathscr{S})}{\log_2 3} \approx 1.678$.

Summary of lecture three

- Shannon codes, used to prove the noiseless coding theorem, are ‘good’ but they are not optimal (i.e., they are not the shortest codes available).
- Huffman coding provides an algorithm to construct optimal codes for any given information source.
- In practice, however, Huffman coding by itself is pretty much useless: much more sophisticated methods are required.

Keywords of lecture three

Shannon codes, ‘good’ codes versus optimal codes, binary and D-ary Huffman codes