Chapter 6

Data Compression

Exercise 1

For a given random variable X with the probability distribution $\mathcal{P}(X = x_i) = p_i$ and corresponding codeword lengths $(l_i)_i$ of a code C we define $L_C^{\diamondsuit}(X) = \sum_i p_i l_i^{100}$. For a fixed random variable X let $L_1^{\diamondsuit}(X) = \min_{C \in \text{prefix}} L_C^{\diamondsuit}(X)$ denotes the minimum of $L_C^{\diamondsuit}(X)$ over all prefix codes and $L_2^{\diamondsuit}(X) = \min_{C \in \text{ud}} L_C^{\diamondsuit}(X)$ denotes the minimum of $L_C^{\diamondsuit}(X)$ over all uniquely decodable codes.

Compare the quantities $L_1^{\diamondsuit}(X)?L_2^{\diamondsuit}(X)$, i.e. replace '?' with proper symbol, e.g. $<, \leq, >, \neq, =, \geq$, *not comparable*, etc.

Answer of exercise 1

Any prefix code is uniquely decodable, therefore, the minimum $L_1^{\diamondsuit}(X)$ is taken over proper subset of uniquely decodable codes giving directly $L_1^{\diamondsuit}(X) \geq L_2^{\diamondsuit}(X)$.

However, from MacMillan and Kraft inequality we have that for any uniquely decodable code C we can construct prefix code C' with the same codeword lengths. Since the quantity $L_C^{\diamondsuit}(X)$ depends only on X and codeword lengths, we have that for any C minimizing $L_2^{\diamondsuit}(X)$ we can construct prefix C' such that $L_C^{\diamondsuit}(X) = L_{C'}^{\diamondsuit}(X)$ giving $L_1^{\diamondsuit}(X) \leq L_2^{\diamondsuit}(X)$.

Together we have $L_1^{\diamondsuit}(X) = L_2^{\diamondsuit}(X)$.

Exercise 2

Let C be a prefix code over the alphabet D ($d = |D|$) with codeword lengths $l_1, l_2, \ldots, l_m$ that satisfy

$$\sum_{i=1}^{m} d^{-l_i} < 1. \tag{6.1}$$

Show that there exists an arbitrarily long sequence in D^* that cannot be decoded into a sequence of codewords.

Answer of exercise 2

Consider the tree from the proof of Kraft inequality. Since $\sum_{i=1}^{m} d^{-l_i} < 1$, we have that there is a path from the root to a leaf of depth l_{max} such that there is no codeword on this path (otherwise all nodes in depth l_{max} are covered and Inequality (6.1) turns into equality). Such a path specifies a sequence $x \in D^*$. It holds that for any $y \in D^*$ and any prefix z of xy there is no codeword to be prefix of z. Note that we can construct z of arbitrary length, what gives the desired result.

Exercise 3

We say that $C : \mathbf{Im}(X) \to D^*$ is a suffix code iff there are no $x_1, x_2 \in \mathbf{Im}(X)$ such that $C(x_1)$ is a suffix of $C(x_2)$.

Show that for any suffix code C over alphabet D (with $|D| = d$), and any random variable X, it holds that

$$L_C(X) \geq H_d(X).$$

Answer of exercise 3

This property holds for prefix codes (see lecture transparencies). It suffices to show that for any suffix code C we may construct prefix code C' such that $L_C(X) \geq L_{C'}(X)$.

Let x^R denotes the reverse of the word $x \in D^*$. Given the suffix code C we define the code $C'(x) = C(x)^R$. It is easy to see that C' is a prefix code. Otherwise there are $x_1, x_2 \in \mathbf{Im}(X)$ such that $C'(x_1) = C'(x_2)w$, where $w \in D^*$. However, this implies that $C(x_1) = C'(x_1)^R = w^R C'(x_2)^R = w^R C(x_2)$ and C is not a suffix code.

The equality $L_C(X) = L_{C'}(X)$ follows directly since $l(C(x)) = l(C(x)^R)$.

Exercise 4

A code C is fix-free iff it is both prefix and suffix code.

Let $l_1, l_2, \ldots, l_m$ be positive integers such that

$$\sum_{k=1}^{m} 2^{-l_k} \leq \frac{1}{2}.$$

Show that there exists a binary fix-free code with codeword lengths $l_1, l_2, \ldots, l_m$.

Is the opposite implication true?

Answer of exercise 4

Let $l_{max} = \max_{k=1,2,\ldots,m} l_k$. Let us consider two full binary trees of depth l_{max}. Vertices of the first one will be labeled by words from D^*, where the correspondence is the same as in the proof of the Kraft inequality (prefix tree). Vertices in the second tree will be labeled in the reverse order, i.e. the codeword will be written from the root to a leaf in the reverse order (suffix tree).

At the beginning all vertices in both trees are marked as 'free'. Let us suppose that WLOG the codeword lengths are ordered as $l_1 \leq l_2 \leq \cdots \leq l_m$. Let us construct the code as follows. Take each l_k sequentially ($k = 1, 2, \ldots, m$) and

- Choose a vertex of depth l_k (in the prefix tree) 'free' both in the prefix and suffix tree. Note that in the prefix tree the vertex is labeled by x_k and in the suffix tree it is labeled by x_k^R (but is in the same depth l_k).
- Mark it as 'codeword' in both trees
- Mark all its successors as 'used' in both trees.

Note that we never select a predecessor of a vertex already labeled as 'codeword' in this way since the codewords are ordered according to their length.

Using such a procedure we can construct only code that is fixfree - it is prefix because codewords have no codeword predecessors and successors in the prefix tree and it is suffix because codewords have no codeword predecessors and successors in the suffix tree.

It remains to verify whether we can always construct such a code, i.e. whether we can always find a vertex of desired depth 'free' both in the prefix and suffix tree. When marking a specific vertex as a 'codeword', we have to label $2^{l_{max}-l_k}$ vertices of depth l_{max} as 'codeword' or 'used' both in prefix and suffix tree. Note that these two sets can be disjoint (their intersection are palindroms), but, when marking a single vertex as 'codeword' we mark at most $2 \times 2^{l_{max}-l_k}$ vertices as 'used' in at least one tree. Therefore, after placing c codewords, we have at least

$$2^{l_{max}} - 2\sum_{k=1}^{c} 2^{l_{max}-l_k}$$

codewords free in both trees.

If we find some vertex in depth l_{max} marked as 'free' both in the prefix and the suffix tree, then all vertices on the path from the root to the leaf are marked as 'free' and we may use them. Finally, we can always find vertex of dept l_{max} free both in prefix and suffix tree, because

$$\sum_{k=1}^{m} 2^{-l_k} \leq \frac{1}{2}$$

giving

$$2\sum_{k=1}^{m} 2^{l_{max}-l_k} \leq 2^{l_{max}}.$$

The opposite implication is not true, because the sets of vertices removed at depth l_{max} from the prefix and suffix tree are not disjoint in general and therefore we can construct fix-free code with codeword lengths exceeding the 1/2 bound. Example of such situation is e.g. if both x and x^R are in C, for some x. Such a pair consumes only one vertex in the prefix tree and one vertex in the suffix tree.

Exercise 5

A random variable X takes on $m = \mathbf{Im}(X)$ values and has entropy $H(X)$. We have a prefix ternary code C for this source with the average length

$$L_C(X) = \frac{H(X)}{\log_2 3}.$$

1. Show that each symbol of $\mathbf{Im}(X)$ has a d-adic probability distribution equal to 3^{-i} for some $i \in \mathbb{N}_0$.
2. Show that m is odd.

Answer of exercise 5

1. We know that the length $L_C(X)$ of any d-ary prefix code satisfies

$$\frac{H(X)}{\log_2 d} = H_d(X) \leq L_C(X)$$

with equality if and only if the probability distribution of X is d-adic, what gives that all probabilities are negative powers of d, in our specific case negative powers of 3.

2. Let $k = \max_{x \in \mathbf{Im}(X)} - \log_3 P(X = x)$. Then

$$1 = \sum_{x \in \mathbf{Im}(X)} P(X = x) = 3^{-k} \sum_{x \in \mathbf{Im}(X)} 3^k P(X = x).$$

$3^k P(X = x)$ is an nonnegative integer power of 3 (because it holds that $P(X = x) \geq 3^{-k}$), thus it is an odd integer. Sum of these integers must equal to 3^k, i.e. it is odd as well. Together we have that the number of summands m must be odd as well.

6.1 Huffman coding

Exercise 6

Consider a random variable X taking values in the set $\mathbf{Im}(X) = \{1, 2, 3, 4, 5\}$ with probabilities $0.25, 0.25, 0.2, 0.15, 0.15$, respectively. Construct an optimal prefix ternary code for this random variable and calculate the average length of the codewords.

Answer of exercise 6

Huffman coding ...

Exercise 7

Prove that if $p_1 > 0.4$, then the shortest codeword of a binary Huffman code has length equal to 1. Then prove that the redundancy of such a Huffman code is lower bounded by $1 - h_b(p_1)$.

Exercise 8

Consider the random variable

X	x_1	x_2	x_3	x_4	x_5	x_6	x_7
$\mathcal{P}(X = x_i)$	0.49	0.26	0.12	0.04	0.04	0.03	0.02

1. Find a binary Huffman code for $\mathbf{X}$.
2. Find the expected code length for this encoding.
3. Find a 5-ary Huffman code for $\mathbf{X}$.

Exercise 9

Find the binary Huffman code for the source with probabilities $(1/3, 1/5, 1/5, 2/15, 2/15)$. Argue whether this code is also optimal for the source with probabilities $(1/5, 1/5, 1/5, 1/5, 1/5)$.

Exercise 10

Which of these codes cannot be Huffman codes for any probability assignment?

1. $\{0, 10, 11\}$.
2. $\{00, 01, 10, 110\}$.
3. $\{01, 10\}$.

Exercise 11

Words like Run! Help! and Fire! are short, not because they are frequently used, but perhaps because time is precious in the situations in which these words are required. Suppose that $[\mathbf{X} = i]$ with probability p_i, $i = 1, 2, \ldots, m$. Let l_i be the length of the binary codeword associated with $[\mathbf{X} = i]$ in a particular code C, and let c_i denote the cost per symbol of the codeword when $[\mathbf{X} = i]$. Thus, the average cost $\mathcal{C}_C(X)$ of the description of $\mathbf{X}$ is

$$\mathcal{C}_C(X) = \sum_{i=1}^{m} p_i c_i l_i.$$

1. Minimize $\mathcal{C}_C(X)$ over all $l_1, l_2, \ldots, l_m$ such that $\sum_{i=1}^{m} 2^{-l_i} \leq 1$. Ignore any implied integer constraints on l_i. Determine $l_1^*, l_2^*, \ldots, l_m^*$ minimizing $\mathcal{C}_C(X)$, and find the associated minimum value $\mathcal{C}^*(X)$.
2. How would you use the Huffman code procedure to minimize $\mathcal{C}_C(X)$ over all uniquely decodable codes? Let $C_{\text{ud}}(X)$ denotes this minimum.
3. Show that
$$\mathcal{C}^*(X) \leq C_{\text{ud}}(X) < \mathcal{C}^*(X) + \sum_{i=1}^{m} p_i c_i.$$

Answer of exercise 11

1. Let $n = \sum_i p_i c_i$. Then the values $\frac{p_i c_i}{n}$ form a probability distribution. Moreover, n does not depend on the choice of l_i, and the quantities $\mathcal{C}_C(X)$ and $\mathcal{C}_C(X)/n$ attain minimum for the same C, i.e. codeword lengths. It remains to minimize
$$\mathcal{C}_C(X)/n = \sum_{i=1}^{m} \frac{p_i c_i}{n} l_i.$$
This situation is analogous to the average length $L_C(X)$. However, in this case the 'price we pay' per each codeword symbol in each codeword is not the original probability p_i of the codeword, but the quantity $c_i p_i / n$. However, it is a probability distribution as well, so all theorems for minimizing the average length apply.
We have that the minimal values are $l_i^* = -\log(c_i p_i / n)$ and the associated minimal value $\mathcal{C}^*(X)$ is the entropy of the probability distribution $(c_i p_i / n)_i$ multiplied by n, i.e.
$$\mathcal{C}^*(X) = nH(N)$$
with N defined by $P(N = i) = c_i p_i / n$.
2. Using the arguments in the Task 1, we construct Huffman code for the distribution N, and have the value
$$C_{\text{ud}}(X) = nL_{\text{Huff}}(N).$$
3. Using the arguments from the Task 1 we know that
$$\frac{\mathcal{C}^*(X)}{n} = H(N) \leq L_{\text{Huff}}(N) = \frac{C_{\text{ud}}(X)}{n} < H(N) + 1 = \frac{\mathcal{C}^*(X)}{n} + 1.$$
We multiply the equation by n to get the desired result
$$\mathcal{C}^*(X) = nH(N) \leq nL_{\text{Huff}}(N) = C_{\text{ud}}(X) < nH(N) + n = \mathcal{C}^*(X) + n.$$

Exercise 12

Although the codeword lengths of an optimal variable length code are complicated functions of the message probabilities $(p_1, p_2, \ldots, p_m)$, it can be said that less probable symbols are encoded into longer codewords. Suppose that the message probabilities are given in decreasing order $p_1 > p_2 \geq \cdots \geq p_m$.

1. Prove that for any binary Huffman code, if the most probable message symbol has the probability $p_1 > 2/5$, then that symbol must be assigned a codeword of length 1.
2. Prove that for any binary Huffman code, if the most probable message symbol has probability $p_1 < 1/3$, then that symbol must be assigned a codeword of length ≥ 2.

6.2 Lempel-Ziv coding

Exercise 13

Compress $(10)^{10}$ using the LZ coding.

Answer of exercise 13

The string is parsed as 1, 0, 10, 101, 01, 010, 1010, 1010. We code it as $(000, 1)$, $(000, 0)$, $(001, 0)$, $(011, 1)$, $(010, 1)$, $(101, 0)$, $(100, 0)$, $(100, 0)$.

Exercise 14

Decode the string 001000101010100 that was encoded using LZ coding.

Answer of exercise 14

Obviously, we can make string human readable by adding parentheses and commas: $(00, 1)$, $(00, 0)$, $(10, 1)$, $(01, 0)$, $(10, 0)$. We start building phrases: 1, 0, 01, 10, 00. We concatenate the phrases 10011000.

Exercise 15

Compress $1^{10}0^{10}$ using the LZ coding.

Answer of exercise 15

The string is parsed as 1, 11, 111, 1111, 0, 00, 000, 0000. We code it as $(000, 1)$, $(001, 1)$, $(010, 1)$, $(011, 1)$, $(000, 0)$, $(101, 0)$, $(110, 0)$, $(111, 0)$.

Exercise 16

Decode the string 000000100100011100011011111011011 that was encoded using LZ coding.

Answer of exercise 16

Obviously, we can make string human readable by adding parentheses and commas: $(000, 0)$, $(001, 0)$, $(010, 0)$, $(011, 1)$, $(000, 1)$, $(101, 1)$, $(110, 1)$, $(101, 1)$. We start building phrases: 0, 00, 000, 0001, 1, 11, 111, 11. We concatenate the phrases $0^9 1^9$.