INFORMATION THEORY
Master of Logic, Master AI, Master CS, University of Amsterdam, 2018
TEACHER: Christian Schaffner, TAs: Yfke Dulek, Esteban Landerreche, Kyah Smaal

Team homework set 3

Problem 1: Huffman Coding

(a) **(4pt)** Consider the binary source P_X with $P_X(0) = \frac{1}{8}$ and $P_X(1) = \frac{7}{8}$.
Design a Huffman code for P_X. Describe, in order, which source symbols you combine, and list the final codebook. What is the average codeword length?

(b) **(4pt)** Design a Huffman code for blocks of $N = 2$ bits drawn from the source P_X. Describe, in order, which source symbols you combine, and list the final codebook. What is the average codeword length?

(c) **(4pt)** Design a Huffman code for blocks of $N = 3$ bits drawn from the source P_X. Describe, in order, which source symbols you combine, and list the final codebook. What is the average codeword length?

(d) **(4pt)** For the three codes you designed ($N = 1, 2, 3$), divide the average codeword length by N, and compare these values to the optimal length, i.e., $H(X)$. What do you observe?

(e) **(1pt)** If you were asked to design a Huffman code for a block of $N = 100$ bits, what problem would you run into?

(f) **(2pt)** Consider the random variable Z with

z	1	2	3	4	5	6
$P_Z(z)$	1/10	3/10	2/10	2/10	1/10	1/10

Design a *ternary* Huffman code for Z (i.e. using an alphabet with three symbols).

Problem 2: Inefficiency when using the wrong code

(a) **(2pt)** Given are two distributions P_X and Q_X for the same set $\mathcal{X} = \{\mathtt{a}, \mathtt{b}, \mathtt{c}, \mathtt{d}\}$:

x	a	b	c	d
$P_X(x)$	1/4	1/4	1/4	1/4
$Q_X(x)$	1/2	1/4	1/8	1/8

Design an optimal prefix-free code for $Q_X(x)$.

(b) **(4pt)** What is the expected codeword length of the code you just designed? (Three decimals precision)

(c) **(4pt)** What is the expected codeword length if you use this code to encode the source P_X? (Three decimals precision)

(d) **(4pt)** Show that in general, when using optimal code for any source Q_X (which has lengths $\ell(x) = \lceil -\log Q_X(x) \rceil$ for all $x \in \mathcal{X}$) to encode another source P_X, this incurs a penalty of $D(P_X||Q_X)$ in the average description length.
More formally, prove that

$$H(P_X) + D(P_X||Q_X) \leq \mathbb{E}_{P_X}[\ell(X)] \leq H(P_X) + D(P_X||Q_X) + 1$$

for all P_X and Q_X.

Problem 3: Shannon code (6pt)

Consider the following method for generating a code for a random variable X which takes on m values $\{1, 2, ..., m\}$. Assume that the probabilities are ordered such that $P_X(1) \geq P_X(2) \geq ... \geq P_X(m)$. Define

$$F_i := \sum_{k=1}^{i-1} P_X(k),$$

the sum of the probabilities of all symbols less than i. Then the Shannon code is defined by assigning the (binary representation of the) number $F_i \in [0, 1]$ as the codeword for i, where F_i is rounded off to $\lceil \log \frac{1}{P_X(i)} \rceil$ bits.

(a) **(1pt)** Construct the code for the probability distribution $P_X(1) = \frac{1}{2}$, $P_X(2) = \frac{1}{4}$, $P_X(3) = P_X(4) = \frac{1}{8}$

(b) **(1pt)** Construct the code for the probability distribution $P_Y(1) = P_Y(2) = P_Y(3) = \frac{1}{3}$.

(c) **(2pt)** Show that the Shannon code is prefix-free.

(d) **(2pt)** Show that the average length ℓ_S of the Shannon code satisfies

$$H(X) \leq \ell_S(P_X) < H(X) + 1.$$

Problem 4: Sampling from any distribution using random bits

In this exercise, we come up with a strategy to sample from an arbitrary distribution P_X using fair random bits (for example, the outcome of a sequence of fair coin tosses).

(a) Let Z_1 be a random variable with $\mathcal{Z}_1 = \{a, b, c\}$ and $P_{Z_1}(a) = 1/2$, $P_{Z_1}(b) = P_{Z_1}(c) = 1/4$. Come up with a strategy to sample from X using a number of fair coin tosses. How many coin tosses do you expect to do? How does this compare to the entropy of Z_1?

(b) Consider the standard binary representation of some $p_i \in [0, 1)$. Let the *atoms* of this representation be the set $At_i := \{2^{-k} \mid$ the k^{th} bit of the binary representation of p_i is $1.\}$. Find the atoms for the binary expansion of $p_1 = \frac{1}{3}$ and $p_2 = \frac{2}{3}$.

(c) Show that for any probability distribution with probabilities $(p_1, ..., p_n)$, it is possible to construct a binary tree (the *sampling tree* for this distribution) such that if $2^{-k} \in At_i$ for some i, then the tree contains a leaf with label i at depth k. **Hint:** use Kraft's inequality.

(d) Let Z_2 be a random variable with $\mathcal{Z}_2 = \{a, b\}$ and $P_{Z_2}(a) = 1/3$, $P_{Z_2}(b) = 2/3$. Construct the sampling tree for P_{Z_2}. Find a fair coin and use it to sample from this distribution, following the strategy described by the sampling tree.

Let $ET(X)$ denote the expected number of coin tosses when sampling from X using the sampling tree described above. In the rest of this exercise, you will show that this method of sampling from an arbitrary distribution P_X using fair random bits is quite efficient in terms of $ET(X)$.

(e) Given a sampling tree for an arbitrary distribution P_X, define a random variable Y with $\mathcal{Y}$ the set of all leafs of the tree, and $P_Y(y) = 2^{-d(y)}$, where $d(y)$ is the depth of the leaf y in the tree. Prove that $H(Y) = ET(X)$.

(f) Use the result from (e) to prove that $H(X) \leq ET(X)$.

★ Prove that $H(Y|X) < 2$ (**Hint:** see Cover and Thomas, Section 5.12)

(g) Use the result from ★ to prove that $ET(X) < H(X) + 2$.

Problem 5: Programming project (10pt)

Invent a compressor and uncompressor for a source file of $N = 10000$ bits each having probability $p = 0.01$ of being a 1. Implement them and estimate how well your method works.

Evaluate the performance of your algorithm on this particular file. Use this program to generate more testdata.