TUTORIAL IV

1 Homework 3

1.1 Huffman's algorithm

Huffman's algorithm constructs a prefix code C_{H} given a distribution $(p_1, \ldots, p_m)$ on the symbols $\{1, \ldots, m\}$. The objective of this problem is to show that the expected length $L(C_{\mathrm{H}})$ is minimum among all the prefix codes. Huffman's algorithm constructs a binary tree as follows. The algorithm starts with independent nodes labeled by the elements $1, \ldots, m$ and the corresponding probability. At the beginning, all the nodes or marked unvisited. At each step, we choose the two unvisited nodes u, v with minimum value of p_u, p_v. We create a new node w with an assigned probability $p_w = p_u + p_v$ which is the parent of u and v. w is marked as unvisited and u, v are marked as visited. The step is repeated $m - 1$ times until we have one unvisited node (the root) with an assigned probability 1. To every path from the root to a leaf of the tree, we assign a bitstring where a "left" edge is read as 0 and a "right" edge is read as 1. The obtained tree defines a code in the following way: for any $x \in \{1, \ldots, m\}$, $C_{\mathrm{H}}(x)$ is the bitstring corresponding to the path from the root to x.

1. Show that for any optimal code, it can be transformed to one with the following property: the two longest codewords correspond to the two least likely symbols, and they have the same length and they only differ in the last bit.

   A: *Given an optimal code, consider the set S of symbols with maximal codeword size. Our first claim is that symbols in S are the ones with the lowest probability. Indeed, if another symbol with strictly lower probability has a shorter codeword, exchanging this symbol with one of S would result in a stricly better code. Hence, the two least likely symbols are in S, and since we can swap elements which are at the same depth, we can put these two element as childs of a same node. This gives us a code with same expected length in which the two least likely symbol have the longest codewords, have the same length, and only differ in the last bit.*

2. Conclude that C_{H} achieves the optimal expected length for $(p_1, \ldots, p_m)$.

   A: *We will show this by induction on m:*

   - *If $m = 2$, then the complete tree of depth 1 is optimal*
   - *If $m > 2$, we take an optimal code for $(p_1, \ldots, p_m)$, and we consider the optimal code C given by the previous procedure. We now merge the two least likely symbol x, y into one new symbol z with the probability $p'(z) = p(x) + p(y)$. Surely, if we consider the code C and merge the two symbol by forgetting the last bit, it gives us a code C' for the new distribution p'. By induction hypothesis, the code C'_H producted by Huffman algorithm for p' is optimal.*

     Now, going back to p, we add $p(x) + p(y)$ to the expected length of both code. Thus, since we had $L(C'_H) \leq L(C')$, we have $L(C_H) \leq L(C)$, but since C is optimal, we have $L(C_H) = L(C)$. Hence the Huffman algorithm gives an optimal coding for a distribution of size m.

1.2 Data compression

This problem is to illustrate that some stream codes can use much less than one bit per symbol of the source. Assume the source is composed of symbols in $\mathcal{X}$ and $0 \in \mathcal{X}$ is one of the symbols. To simplify the calculations, you may assume $\mathcal{X} = \{0, 1\}$. For the two types of encodings described below, determine the length of the encoding of the bitstring 0^n (i.e., n times the symbol 0). You can give your result in the form $O(f(n))$ for the smallest possible f.

1. Arithmetic code with the following simple probabilistic model: $P_\ell(x_\ell|x_1\dots x_{\ell-1}) = \frac{|\{i\in\{1,\dots,\ell-1\}:x_i=x_\ell\}|+1}{\ell-1+|\mathcal{X}|}$.

   A: *We set $p=|\mathcal{X}|$, and we consider the first steps to understand how the encoding will work:*

   - *For the first symbol, the outcomes are equiprobable, hence if we want to specify the outcome 0, we have to take a fraction inside the interval $\left[0;\frac{1}{p}\right)$.*
   - *For the second symbol, the outcome 0 is more likely since:*

   $$\mathbf{P}_{x_1|x_0=0}(0) = \frac{2}{p+1} > \frac{1}{p+1} = \mathbf{P}_{x_1|x_0=0}(i) \quad \forall i \geq 0$$

   Thus the interval became $\left[0;\frac{1}{p}\frac{2}{p+1}\right)$.

   - *As n grows, the outcome 0 become more and more likely:*

   $$\mathbf{P}_{x_n|x_0=\dots=x_{n-1}=0}(0) = \frac{n+1}{p+n} > \frac{1}{p+n} = \mathbf{P}_{x_1|x_0=\dots=x_{n-1}=0}(i) \quad \forall i \geq 0$$

   Thus, the interval corresponding to 0^n is:

   $$\left[0;\prod_{i=0}^{n-1}\frac{i+1}{p+i}\right) = \left[0;\frac{n!(p-1)!}{(p+n-1)!}\right) = \left[0;\frac{1}{\binom{n+p-1}{n}}\right)$$

   Using Stirling approximation, we have:

   $$\binom{n+p-1}{n} \sim 2^{(n+p-1)\log(n+p-1)-n\log n-(p-1)\log(p-1)}$$

   The number $L(n)$ of bits to describe a fraction in this interval is:

   $$L(n) = \log\binom{n+p-1}{n} \sim c + (n+p-1)\log(n+p-1) - n\log n$$

   After some simplifications, we find:

   $$L(n) \sim c' + (p-1)\log(n+p-1) \sim (p-1)\log(n)$$

   Thus, we have $L(n) = O(\log n)$.

2. There is a widely used family of compression algorithms named after Lempel-Ziv. Unlike all schemes we have seen in class that explicitly use the probabilistic model, these compression algorithms do not make use of any probabilistic model and they are called *universal* for this reason. The basic idea is to use repetitions of sequences of symbols. See http://www-math.mit.edu/ shor/PAM/lempel_ziv_notes.pdf for a description of a simple variant that you are asked to consider.[1]

   A: *Since the input strings is only made of 0, the Lempel-Ziv algorithm will cut it into the following pieces:*

   $$0|00|000|0000|00000|\dots|00\dots0$$

   The first phrase in the dictionnary is 0, of size 1, then 00, of size $2,\dots$ In this particular case, each phrase is used at most one (except for the end of the string), thus the total length n is the sum of the size of the phrases. Since we have $\sum_{i=0}^{n} i = \frac{n(n-1)}{2}$, the number of phrases is thus $O(\sqrt{n})$.

   A first over approximation of the number of symbols $L(n)$ is given by the following remark: if there are $r(n)$ phrases, we used at most $\log r(n) + \log 2$ bits to encode each of them, so the upper bound is $r(n)(\log r(n)+1)$. In our settings:

   $$L(n) = O(\sqrt{n}\times(\log\sqrt{n}+1)) = O(\sqrt{n}\log n)$$

[1] The linked document also shows that if one applies this compression algorithm to an i.i.d. sequence of symbols, the number of used bits per symbol is optimal: it is the entropy of the source.

2 Code for unknown distribution

Recall that we can build a code C that achieve an expected description length $l(C)$ within 1 bit of the lower bound, that is:

$$H(X) \leq l(C) < H(X) + 1$$

This is done using the following choice of word lengths: $l_i = \left\lceil \log \frac{1}{p_i} \right\rceil$. In some case, we don't know the true distribution p, but only have an approximation q, and still want to find a code.

1. Show that if we use the same choice of word lengths: $l_i = \left\lceil \log \frac{1}{q_i} \right\rceil$, we have:

$$H(p) + D(p||q) \leq E_p(l(C)) < H(p) + D(p||q) + 1$$

A: *The expected codelength is:*

$$\begin{aligned}
E_p(l(X)) &= \sum_x p(x) \left\lceil \log \frac{1}{q(x)} \right\rceil \\
&< \sum_x p(x) \left(\log \frac{1}{q(x)} + 1 \right) \\
&= \sum_x p(x) \left(\log \frac{1}{p(x)} \frac{p(x)}{q(x)} + 1 \right) \\
&= \sum_x p(x) \log \frac{1}{p(x)} + \sum_x p(x) \log \frac{p(x)}{q(x)} + \sum_x p(x) \\
&= H(p) + D(p||q) + 1
\end{aligned}$$

The lower bound can be derived in a similar way.

3 Channel capacity

1. For a discrete channel $W_{\mathcal{Y}|\mathcal{X}}$ with input alphabet $\mathcal{X}$, output alphabet $\mathcal{Y}$ and probability transition matrix $p(y|x)$, let Let $C(W)$ denote the channel capacity of W. Show that
   (a) $C(W) \geq 0$.
   A: $C(W) = \max_{p(x)} I(X;Y) \geq 0$ *since* $I(X;Y) \geq 0$.
   (b) $C(W) \leq \log_2 |\mathcal{X}|$.
   A: $C(W) = \max_{p(x)} I(X;Y) = \max_{p(x)} H(X) - H(X|Y) \leq \max_{p(x)} H(X) = \log_2 |\mathcal{X}|$.
   (c) $C(W) \leq \log_2 |\mathcal{Y}|$.
   A: *Similar to (b) because* $I(X;Y)$ *is symmetric in* X *and* Y.
   (d) $I(X;Y)$ is a continuous concave function of $p(x)$.
   A: $I(X;Y) = H(Y) - H(Y|X) = H(Y) - \sum_x p(x) H(Y|X=x)$. *We know that* $H(Y)$ *is a continuous concave function of* $p(y)$. *Since* $p(y|x)$ *is fixed,* $p(y)$ *is a linear function of* $p(x)$ *and so,* $H(Y)$ *is a continuous concave function of* $p(x)$. *The second term (the sum) is a linear function of* $p(x)$. *Hence the difference is a continuous concave function in* $p(x)$.

2. Given a channel $W_{\mathcal{X}|\mathcal{Y}}$ with transition probabilities $p(y|x)$ and channel capacity $C(W) = \max_{p(x)} I(X;Y)$, suppose you apply a preprocessing step to the output by forming $\tilde{Y} = g(Y)$.
   (a) Does it strictly improve the channel capacity?

A: *Here, Y depends on X and $\tilde{Y}$ depends only on Y and so $X \to Y \to \tilde{Y}$ forms a Markov chain. We have $I(X;Y,\tilde{Y}) = I(X;\tilde{Y}) + I(X;Y|\tilde{Y}) = I(X;Y) + I(X;\tilde{Y}|Y)$. Since X and $\tilde{Y}$ are independent conditioned on Y, $I(X;\tilde{Y}|Y) = 0$ and hence $I(X;Y) > I(X;\tilde{Y})$ for every distribution $p(x)$ on $\mathcal{X}$. Let $\tilde{p}(x)$ be the distribution for which $I(X;\tilde{Y})$ is maximised and let $\tilde{C}(W) = \max_{p(x)} I(X;\tilde{Y}) = I_{\tilde{p}(x)}(X;\tilde{Y})$. We have*

$$C(W) = \max_{p(x)} I(X;Y) \geq \max_{\tilde{p}(x)} I(X;Y) \geq I_{\tilde{p}(x)}(X;\tilde{Y}) = \tilde{C}(W).$$

Therefore, the preprocessing step does not improve the channel capacity.

(b) Under what conditions does the capacity not strictly decrease?

A: *$C(W) = \tilde{C}(W)$ if $X \to \tilde{Y} \to Y$ forms a Markov chain (i.e., when X and Y are conditionally independent given $\tilde{Y}$ or $I(X;Y|\tilde{Y}) = 0$).*

4 Jointly typical sequences

The set $A_\epsilon^{(n)}$ of jointly typical sequences $\{(x^n, y^n)\}$ with respect to the distribution $p_{XY}(x,y)$ is given by

$$A_\epsilon^{(n)} = \left\{ (x^n, y^n) \in \mathcal{X}^n \times \mathcal{Y}^n : \left| -\frac{1}{n} \log p_{X^n}(x^n) - H(X) \right| < \epsilon, \right.$$
$$\left| -\frac{1}{n} \log p_{Y^n}(y^n) - H(Y) \right| < \epsilon,$$
$$\left. \left| -\frac{1}{n} \log p_{X^nY^n}(x^n, y^n) - H(X,Y) \right| < \epsilon \right\}$$

where $p_{X^nY^n}(x^n, y^n) = \prod_{i=1}^n p_{XY}(x_i, y_i)$.

1. Let (X^n, Y^n) be sequences of length n drawn i.i.d. according to $p_{X^nY^n}(x^n, y^n) = \prod_{i=1}^n p_{XY}(x_i, y_i)$. Then show that

   (a) $\Pr[(X^n, Y^n) \in A_\epsilon^{(n)}] \to 1$ as $n \to \infty$.

   A: *By weak law of large numbers, we have $-\frac{1}{n} \log p_{X^n}(X^n) \to -\mathrm{E}[\log p_X(X)] = H(X)$ and hence for $\epsilon > 0$, there exists n_1 such that for all $n > n_1$,*

   $$\Pr\left[\left| -\frac{1}{n} \log p_{X^n}(X^n) - H(X) \right| \geq \epsilon \right] < \frac{\epsilon}{3}.$$

   Similarly, there exist n_2 and n_3 such that for all $n \geq n_2$,

   $$\Pr\left[\left| -\frac{1}{n} \log p_{Y^n}(Y^n) - H(Y) \right| \geq \epsilon \right] < \frac{\epsilon}{3}$$

   and for all $n \geq n_3$,

   $$\Pr\left[\left| -\frac{1}{n} \log p_{X^nY^n}(X^n, Y^n) - H(X,Y) \right| \geq \epsilon \right] < \frac{\epsilon}{3}.$$

   Choose $n_0 = \max\{n_1, n_2, n_3\}$. For $n \geq n_0$, the probability of the union of the above three events should be less than ϵ. Hence $\Pr[(X^n, Y^n) \notin A_\epsilon^{(n)}] \to 0$ as $n \to \infty$.

   (b) $|A_\epsilon^{(n)}| \leq 2^{nH(X,Y)+\epsilon}$.

   A: *We have,*

   $$1 = \sum p_{X^nY^n}(x^n, y^n) \geq \sum_{(x^n,y^n) \in A_\epsilon^{(n)}} p_{X^nY^n}(x^n, y^n) \geq |A_\epsilon^{(n)}| 2^{-n(H(X,Y)-\epsilon)},$$

   and hence $|A_\epsilon^{(n)}| \leq 2^{n(H(X,Y)-\epsilon)}$.

(c) If $(\tilde{X}^n, \tilde{Y}^n) \sim p_{X^n}(x^n)p_{Y^n}(y^n)$, then $\Pr[(\tilde{X}^n, \tilde{Y}^n) \in A_\epsilon^n] \leq 2^{-n(I(X;Y)-3\epsilon)}$.

A:

$$\begin{aligned}
\Pr[(\tilde{X}^n, \tilde{Y}^n) \in A_\epsilon^n] &= \sum_{(x^n,y^n)\in A_\epsilon^n} p_{X^n}(x^n)p_{Y^n}(y^n) \\
&\leq \sum_{(x^n,y^n)\in A_\epsilon^n} 2^{n(\epsilon-H(X))}2^{n(\epsilon-H(Y))} \\
&\leq 2^{n(\epsilon+H(X,Y))}2^{n(\epsilon-H(X))}2^{n(\epsilon-H(Y))} \\
&= 2^{-n(I(X;Y)-3\epsilon)}
\end{aligned}$$