Homework I: Universal compression

(due November 2nd, before tutorial, submit via portail des études only and call your file "Lastname.pdf")

The objective of this homework is to study universal compressors, i.e., that do not use prior knowledge about the source but still achieve close to optimal performance on some classes of sources. We fix some finite alphabet $\mathcal{X}$ of symbols.

1 Types

We start with some basic observations related to types.

1. Let $x^n = x_1 \dots x_n \in \mathcal{X}^n$. The *type* $\hat{P}_{x^n}$ of x^n is defined as the empirical distribution defined by the sequence x^n. More precisely, $\hat{P}_{x^n}(a) = \frac{|\{i \in [n] : x_i = a\}|}{n}$. For $\mathcal{X} = \{a, b, c\}$, what is the type of the string $x^5 = aaabc$? Give a different string that has the same type as x^5.
   We have that $\hat{P}_{x^5} = (3/5, 1/5, 1/5)$. Another sequence with the same type is for instance $aabca$.

2. Let $\mathcal{T}_{\mathcal{X},n}$ be the set of all types for strings in $\mathcal{X}^n$. Show that $|\mathcal{T}_{\mathcal{X},n}| \leq (n+1)^{|\mathcal{X}|-1}$.
   There are $n+1$ choices at most for each entry $\hat{P}_{x^n}(a)$ (since $|\{i \in [n] : x_i = a\}| \in \{0, 1, \dots, n\}$), the last one is determined by the fact that sum is one.

3. For a given type $\hat{P} \in \mathcal{T}_{\mathcal{X},n}$, the *type class* is defined by $T(\hat{P}) = \{x^n \in \mathcal{X}^n : \hat{P}_{x^n} = \hat{P}\}$. For $\mathcal{X} = \{a, b\}$ and an arbitrary n, $0 \leq k \leq n$ and $\hat{P}(a) = \frac{k}{n}$ and $\hat{P}(b) = 1 - \frac{k}{n}$. What is the size $|T(\hat{P})|$ of the type class of $\hat{P}$?
   An element in $T(\hat{P})$ is a defined by a subset of k indices i such that $x_i = a$, so $|T(\hat{P})| = \binom{n}{k}$.

   For more general $\mathcal{X}$, give an expression for the $|T(\hat{P})|$ using multinomial coefficients.
   Similarly, an element of $T(\hat{P})$ is a partition in subsets of size $k_{a_1}, k_{a_2}, \dots, k_{a_{|\mathcal{X}|}}$ of indices, where $k_{a_i} = n\hat{P}(a_i)$. The number of such partitions is given by the multinomial coefficient $\binom{n}{k_{a_1}, k_{a_2}, \dots, k_{a_{|\mathcal{X}|}}}$.

4. Prove that for any $\hat{P} \in \mathcal{T}_{\mathcal{X},n}$, we have $|T(\hat{P})| \leq 2^{nH(\hat{P})}$. *Hint:* Introduce $X^n = X_1 \dots X_n$ to be iid with distribution $\hat{P}$, and for $x^n \in T(\hat{P})$, compute $P_{X^n}(x^n)$.
   We have $P_{X^n}(x^n) = \hat{P}(a_1)^{n\hat{P}(a_1)} \cdots \hat{P}(a_{|\mathcal{X}|})^{n\hat{P}(a_{|\mathcal{X}|})} = 2^{-\sum_i \hat{P}(a_i) \log \hat{P}(a_i)} = 2^{-nH(\hat{P})}$. Then $1 \geq \sum_{x^n \in T(\hat{P})} P_{X^n}(x^n) = |T(\hat{P})| 2^{-nH(\hat{P})}$ and thus $|T(\hat{P})| \leq 2^{nH(\hat{P})}$.

2 Universal compression using types

Define the distribution Q on $\mathcal{X}^n$ where $Q(x^n) = \frac{1}{c_n |T(\hat{P}_{x^n})|}$ where c_n is chosen so that Q is a distribution.

1. Show that $c_n \leq (n+1)^{|\mathcal{X}|-1}$.
   We have $1 = \sum_{x^n} Q(x^n) = \sum_{\hat{P} \in \mathcal{T}_n} \sum_{x^n \in T(\hat{P})} \frac{1}{c_n |T(\hat{P}_{x^n})|} = \sum_{\hat{P} \in \mathcal{T}_n} \frac{1}{c_n} \sum_{x^n \in T(\hat{P})} \frac{1}{|T(\hat{P})|} = \sum_{\hat{P} \in \mathcal{T}_n} \frac{1}{c_n} = \frac{|\mathcal{T}_n|}{c_n}$. So $c_n \leq |\mathcal{T}_n| \leq (n+1)^{|\mathcal{X}|-1}$.

2. Use Kraft's inequality to show that there is a prefix-free compressor $C_{\text{types}} : \mathcal{X}^n \to \{0,1\}^*$ that maps any $x^n \in \mathcal{X}^n$ to a bitstring $C_{\text{types}}(x^n)$ of length at most $nH(\hat{P}_{x^n}) + (|\mathcal{X}| - 1) \log_2(n+1) + 1$. Note that this compressor does not use any prior model about the source that it is compressing, Q is only a distribution that we use in order to define the compressor and might be different from the actual distribution of the source.
   Set $\ell(x^n) = \lceil \log_2(1/Q(x^n)) \rceil$. It satisfies Kraft's inequality since:

$$\sum_{x^n \in \mathcal{X}_n} 2^{-\lceil \log_2(1/Q(x^n)) \rceil} \leq \sum_{x^n \in \mathcal{X}_n} 2^{-\log_2(1/Q(x^n))} = \sum_{x^n \in \mathcal{X}_n} Q(x^n) = 1$$

Thus, it defines a prefix-free compressor $C_{\text{types}} : \mathcal{X}^n \to \{0,1\}^*$ with $|C_{\text{types}}(x^n)| = \ell(x^n)$. But

$$\begin{aligned}\ell(x^n) &= \lceil \log_2(1/Q(x^n)) \rceil \le \log_2(1/Q(x^n)) + 1 = \log_2 |T(\hat{P}_{x^n})| + \log_2 c_n + 1 \\ &\le nH(\hat{P}_{x^n}) + (|\mathcal{X}| - 1)\log_2(n+1) + 1 \,.\end{aligned}$$

3. Now in order to evaluate how this compression method consider a sequence of independent random variables $X_1 \dots X_n$ each with distribution μ on $\mathcal{X}$. As we saw in class, an optimal-prefix free for this source has expected encoding length between $nH(\mu)$ and $nH(\mu) + 1$ (note that $H(X_1 \dots X_n) = nH(\mu)$).

   (a) Show that for any $x^n \in \mathcal{X}^n$, $|C_{\text{types}}(x^n)| - \log_2 \frac{1}{\mu^n(x^n)} \le (|\mathcal{X}| - 1)\log(n+1) + 1$, where $\mu^n(x^n) := \mu(x_1)\mu(x_2)\cdots\mu(x_n)$.

   $$|C_{\text{types}}(x^n)| - \log \frac{1}{\mu^n(x^n)} \le \log \mu^n(x^n)|T(\hat{P}_{x^n})| + (|\mathcal{X}| - 1)\log(n+1) + 1$$

   But $\mu^n(x^n)|T(\hat{P}_{x^n})| = \sum_{y^n \in T(\hat{P}_{x^n})} \mu^n(y^n) \le 1$.

   (b) Conclude that for any choice of μ, the expected encoding length $\mathop{\mathbf{E}}_{X_1,\dots,X_n \text{ iid } \mu}\{|C_{\text{types}}(X_1 \dots X_n)|\}$ for the universal compressor C_{types} uses at most $O(\log n)$ extra bits compared to the optimal prefix-free code for this source.
   Since $\mathbb{E}\left[\log \frac{1}{\mu^n(X_1,\dots,X_n)}\right] = \sum_{i=1}^n \mathbb{E}\left[-\log \mu(X_i)\right] = nH(\mu)$, we have:

   $$\mathbb{E}[|C_{\text{types}}(X_1,\dots,X_n)|] \le \mathbb{E}\left[\log \frac{1}{\mu^n(X_1,\dots,X_n)}\right] + (|\mathcal{X}|-1)\log(n+1)+1 = nH(\mu)+(|\mathcal{X}|-1)\log(n+1)+1 \,.$$

4. One could then consider more general sources $X_1 \dots X_n$ which are not necessarily iid but define a Markov chain, i.e., we have that

$$P_{X_1 \dots X_n}(x_1 \dots x_n) = \mu(x_1|x_n)\mu(x_2|x_1)\mu(x_3|x_2)\cdots\mu(x_n|x_{n-1}) \tag{1}$$

   for some conditional distribution $\mu(.|.)$ (note that we used a nonstandard circular dependency to avoid the notational complications due to borders). We now define $\hat{P}^2_{x^n}(a,b) = \frac{|\{i \in [n] : x_i = a, x_{i+1} = b\}|}{n}$ (with $x_{n+1} = x_1$) and the type class $T^2(\hat{P}^2) = \{x^n : \hat{P}^2_{x^n} = \hat{P}^2\}$. Construct a universal compressor C^2_{types} as in the previous questions such that for any source of the form (1), C^2_{types} uses at most $O(\log n)$ extra bits compared to the optimal prefix-free code for this source.
   For the sake of simplicity, let us assume that P is a probability distribution. As before, define $Q(x^n) = \frac{1}{d_n |T^2(\hat{P}^2_{x^n})|}$ with d_n parametrized such that Q is a probability distribution. Thus as before $d_n \le |\mathcal{T}^2_n| \le (n+1)^{|\mathcal{X}|^2-1}$ with $\mathcal{T}^2_n$ defined as the set of all types T^2 for strings of length n. Note that $P(x^n) = \prod_{(a,b)} \mu(a|b)^{n\hat{P}^2_{x^n}(a,b)}$ which depends only on the type $\hat{P}^2$ of x^n, and so $|T^2(\hat{P}^2_{x^n})|P(x^n) = \sum_{x^n \in T^2(\hat{P}^2_{x^n})} P(x^n) \le 1$. As such $\log |T^2(\hat{P}^2_{x^n})| - \log 1/P(x_1 \dots x_n) = nH(\hat{P}^2_{x^n}) - \log 1/P(x_1 \dots x_n) \le 1$ and we can have the same argument as before choosing $|C^2_{\text{types}}(x^n)| = \lceil \log_2(1/Q(x^n)) \rceil \le nH(\hat{P}^2{}_{x^n}) + (|\mathcal{X}|-1)\log_2(n+1)+1$ which satisfies Kraft inequality.

   A similar argument works for any finite order Markov chain where X_k is independent of $X_1 \dots X_{k-r-1}$ conditioned on $X_{k-1} \dots X_{k-r}$ (you are not asked to prove this).

3 Universal compression using arithmetic coding

Note that a priori, the compression method using types is not efficient in terms of n. In fact, we used Kraft's inequality to construct a code that has a number of codewords that is exponential in n. In this section, we describe a universal compressor that uses arithmetic coding. Note that nothing prevents us from using arithmetic coding for the choice of distribution Q defined in Section 2, but the issue is that it is not clear that Q satisfies the properties that make arithmetic coding efficient, namely being able to compute conditional probabilities for the next symbol. Here, given a word $x^n \in \mathcal{X}^n$ and $t \in [n]$, define for any $a \in \mathcal{X}$:

$$R_t(a) := \frac{|\{i \in [t-1] : x_i = a\}| + \frac{1}{2}}{t - 1 + \frac{|\mathcal{X}|}{2}} . \tag{2}$$

1. Check that R_t is a probability distribution.
   $\sum_{a\in\mathcal{X}} R_t(a) = \frac{\sum_{a\in\mathcal{X}} |\{i\in[t-1]:x_i=a\}|+\frac{|\mathcal{X}|}{2}}{t-1+\frac{|\mathcal{X}|}{2}} = \frac{t-1+\frac{|\mathcal{X}|}{2}}{t-1+\frac{|\mathcal{X}|}{2}} = 1.$

2. Using R_t for the conditional distributions used for arithmetic coding (in the lectures notes this was denoted $Q_{A_t|A_1\dots A_{t-1}}$), describe a universal compressor C_{arithm} for sequences in $\mathcal{X}^n$ (no need to get into the details of how arithmetic coding works, you are only asked to give a sketch that allows you to discuss the running time of the algorithm). Show that the running time and memory for encoding a sequence is linear in n, assuming $\mathcal{X}$ is constant and the involved arithmetic operations take constant time.
   Let us recall how Arithmetic Coding works. We encode first x^n into an interval $I(x^n) = [u_n, v_n) \subseteq [0, 1)$ and then into $C_{\text{arithm}}(x^n)$.
   First index the symbols $\mathcal{X} = \{a_1, \dots, a_{|\mathcal{X}|}\}$ in an arbitrary way. Then define $I(\emptyset) = [0, 1)$.
   Given $I(x^{i-1}) = [u_{i-1}, v_{i-1})$, we compute $I(x^i) = [u_i, v_i)$ as follows: $x_i = a_k$ for some $k \in [|\mathcal{X}[]$.
   Then the interval is defined in terms of $\alpha_{i,k} = \sum_{p=1}^{k-1} R_i(a_p)$ and $\beta_{i,k} = \sum_{p=1}^{k} R_i(a_p)$ by
   $u_i = u_{i-1} + (v_{i-1} - u_{i-1})\alpha_{i,k}$ and $v_i = u_{i-1} + (v_{i-1} - u_{i-1})\beta_{i,k}$.
   We precompute first all the values of $R_t(a_k)$ in time and space $O(n)$ (with $R_t(a_k) = \frac{\left(t-2+\frac{|\mathcal{X}|}{2}\right)R_{t-1}(a_k)+\mathbf{1}_{x_{t-1}=a}}{t-1+\frac{|\mathcal{X}|}{2}}$),
   and then the $2|\mathcal{X}| \times n$ partial sums $\alpha_{i,k}$ and $\beta_{i,k}$ in time and space $O(n)$. Then, $C_{\text{arithm}}(x^n) = \frac{\lceil 2^\ell u_n \rceil}{2^\ell}$ with $\ell \in \mathbb{N}$ the smallest integer satisfying $\frac{\lceil 2^\ell u_n \rceil + 1}{2^\ell} \le v_n$ can be computed in constant time using logarithm. We have also that $|C_{\text{arithm}}(x_n)| \le \lceil -\log_2(R_1(x_1)R_2(x_2)\cdots R_n(x_n)) \rceil + 1$.

3. For this question, to simplify the calculations, we will assume that $|\mathcal{X}| = 2$. Consider an iid source $X_1 \dots X_n$, where each X_i has distribution μ. Our objective is to show that $|C_{\text{arithm}}(x^n)| - \log_2 \frac{1}{\mu^n(x^n)} \le \frac{1}{2}\log n + O(1)$, where as before $\mu^n(x^n) = \prod_{i=1}^n \mu(x_i)$.

   (a) Show that $R_1(x_1)R_2(x_2)\cdots R_n(x_n) = \frac{\left(\left(n_0-\frac{1}{2}\right)\left(n_0-\frac{3}{2}\right)\cdots\frac{1}{2}\right)\left(\left(n_1-\frac{1}{2}\right)\left(n_1-\frac{3}{2}\right)\cdots\frac{1}{2}\right)}{n!}$, where $n_b = |\{i \in [n] : x_i = b\}|$.

$$\prod_{i=1}^n R_i(x_i) = \prod_{i=1}^n \frac{|\{j \in [i-1] : x_j = x_i\}| + \frac{1}{2}}{i}$$
$$= \frac{1}{n!}\left(\prod_{i\in Z}\left(|\{j \in [i-1] : x_j = 0\}| + \frac{1}{2}\right)\prod_{i\in O}\left(|\{j \in [i-1] : x_j = 1\}| + \frac{1}{2}\right)\right),$$

   where $Z = \{i \in [n] : x_i = 0\}$ is of size n_0 and $O = \{i \in [n] : x_i = 1\}$ is of size n_1. If i is the k-th element of A, we have that $|\{j \in [i-1] : x_j = 0\}| = k - 1$, since by definition of A and i there are

$k-1$ elements $x_j = 0$ before i. The same arises for O and ones. Thus we get:

$$\prod_{i=1}^{n} R_i(x_i) = \frac{1}{n!}\left(\prod_{k=1}^{n_0}\left(k-1+\frac{1}{2}\right)\prod_{k=1}^{n_1}\left(k-1+\frac{1}{2}\right)\right)$$
$$= \frac{\left((n_0-\frac{1}{2})(n_0-\frac{3}{2})\cdots\frac{1}{2}\right)\left((n_1-\frac{1}{2})(n_1-\frac{3}{2})\cdots\frac{1}{2}\right)}{n!}.$$

(b) Show that for $x^n \in \mathcal{X}^n$ and any distribution μ, we have $\mu^n(x^n) \le \frac{n_0^{n_0} n_1^{n_1}}{n^n}$.
We have $\mu^n(x^n) = \mu(0)^{n_0}\mu(1)^{n_1} = 2^{n_0 \log \mu(0) + n_1 \log \mu(1)} = 2^{n(n_0/n \log \mu(0) + n_1/n \log \mu(1))}$.
We can write $(n_0/n \log \mu(0) + n_1/n \log \mu(1)) = (n_0/n \log n_0/n + n_1/n \log n_1/n) - (n_0/n \log n_0/n + n_1/n \log n_1/n) + (n_0/n \log \mu(0) + n_1/n \log \mu(1))$.
With this, $\mu^n(x^n) = 2^{-nH(n_0/n)} 2^{-nD(n_0/n\|\mu)}$ and by nonnegativity of $D(\|)$ it is maximal when $\mu = n_0/n$, where $\mu^n(x^n) = \frac{n_0^{n_0} n_1^{n_1}}{n^n}$.

(c) Conclude. You may use Stirling's approximation $\sqrt{2\pi} n^{n+\frac{1}{2}} e^{-n} \le n! \le e n^{n+\frac{1}{2}} e^{-n}$ for all positive integers. *Remark:* You will get partial credit if you analyze the simpler choice $R'_t(a) = \frac{|\{i \in [t-1] : x_i = a\}|+1}{t+1}$, which gives a slightly worse bound of $\log n$ instead of $\frac{1}{2}\log n$.

$$|C_{\text{arithm}}(x^n)| - \log_2 \frac{1}{\mu^n(x^n)} \le \log_2 \frac{\mu^n(x^n)}{R_1(x_1)R_2(x_2)\cdots R_n(x_n)} + O(1)$$
$$\le \log_2 \frac{n_0^{n_0} n_1^{n_1} n!}{n^n \left((n_0-\frac{1}{2})(n_0-\frac{3}{2})\cdots\frac{1}{2}\right)\left((n_1-\frac{1}{2})(n_1-\frac{3}{2})\cdots\frac{1}{2}\right)} + O(1)$$
$$= \log_2 \frac{n!}{n^n} + \log_2 \frac{n_0^{n_0} n_1^{n_1}}{\frac{(2n_0)!}{2^{2n_0} n_0!}\frac{(2n_1)!}{2^{2n_1} n_1!}} + O(1)$$
$$\le \log_2(e^{-n+1}\sqrt{n}) + \log_2(C e^{n_0} e^{n_1}) + O(1) \quad \text{for some constant } C \text{ using Stirling.}$$
$$= \log_2(eC\sqrt{n}) + O(1) = \frac{1}{2}\log n + O(1)$$