Agenda for the lecture

- Variable-length universal compression scheme 1: A type based scheme (dictionary method)
- Variable-length universal compression scheme 2: Adaptive probability estimation based scheme (entropy method)

19.1 Scheme 1: Type based scheme

Our variable-length universal compression first scheme simply goes over the entire sequence and computes its type. The compressed code consists of type in the header and the specific enumeration of the sequence within the type class as stored as a binary vector. Such methods, where a frequency histogram of the symbols is used for compression are referred to as dictionary based methods in the data compression literature. Schemes such as arithmetic coding which rely on entropy densities instead are referred to as entropy based methods. However, this is practitioner’s nomenclature and at the level of the theoretical treatment that we have seen in this course, the two methods are closely related – type class, too, consists of sequences with constant entropy density.

Formally, our compression protocol is as follows:

*©Himanshu Tyagi. Feel free to use with acknowledgement.
**Input:** A sequence $x \in \mathcal{X}^n$

**Output:** A binary codeword sequence $c(x) \in \{0, 1\}^*$

1. Enumerate all the types using $l_1 = \lceil \log |T| \rceil$ bits. Let $y_1(Q)$ denote the binary vector of length $l_1$ representing the type $Q$.

2. For a type $Q \in T$, enumerate all the sequences of type $Q$ using $l_2(Q) = \lceil \log |T_Q| \rceil$ bits. Let $y_2(x)$ denote the binary vector of length $l_2(Q)$ representing a sequence $x \in T_Q$.

3. Store the given sequence $x$ as $c(x) = (y_1(P_x), y_2(x))$.

The length $l(x)$ of the codeword $c(x)$ is bounded above as

$$l(x) \leq l_1 + l_2(P_x) \leq (|\mathcal{X}| - 1) \log(n + 1) + nH(P_x) + 2,$$

where the last step uses Lemma 18.2 and Lemma 18.3. Therefore, the regret for this code is bounded as

$$r_n(C) = \max_{P^n} \sum_x P^n(x)l(x) - nH(P) \leq (|\mathcal{X}| - 1) \log(n + 1) + n \max_{P^n} \sum_x P^n(x)H(P_x) - nH(P) + 2$$

$$= (|\mathcal{X}| - 1) \log(n + 1) + n[\mathbb{E}_{P^n}[H(P_X)] - H(P)] + 2.$$

Note that for each $a \in \mathcal{X}$,

$$\mathbb{E}_{P^n}[N(a|X^n)] = \mathbb{E}_{P^n}\left[\sum_{i=1}^n \mathbb{1}(X_i = a)\right] = \sum_{i=1}^n \mathbb{E}_{P^n}[\mathbb{1}(X_i = a)] = nP(a).$$

Therefore, $\mathbb{E}_{P^n}[P_{X^n}] = P$, which further gives

$$\mathbb{E}_{P^n}[H(P_X)] \leq H(P),$$
since $H(\cdot)$ is a concave function. Therefore, the regret for the code above satisfies

$$r_n(C) \leq (|X| - 1) \log(n + 1) + 2.$$ 

In fact, a more careful analysis can be used to show that

$$r_n(C) \lesssim \frac{(|X| - 1)}{2} \log n,$$

which in turn yields

$$r^*_n \lesssim \frac{(|X| - 1)}{2} \log n.$$ 

This last bound can be shown to be optimal (we don’t have the tools to complete the course proof at this point). Therefore, the scheme above is almost optimal.

### 19.2 Scheme 2: Add-1 estimator based scheme

The shortcoming of the scheme above is that, first, we need to go over the entire sequence once before we can compress any symbol, and, second, decoding and encoding will require us to operate with a look-up table of exponential size. We already saw a solution which over-came both these shortcomings, namely arithmetic coding. However, it required as an input the estimates of probabilities for each symbol. In particular, if we input $Q(x_i|x_1, \ldots, x_{i-1})$, an arithmetic encoder will produce a codeword of length $l(x) = [-\log Q(x)]$. As saw in the previous lecture, for such a code construction,

$$r_n(C, P^n) \leq D(P^n||Q) + 1.$$ 

Therefore, to bound the regret for this scheme, it suffices to bound

$$\max_{P^n} D(P^n||Q).$$
We now provide a useful choice of $Q \in \mathcal{P}(\mathcal{X}^n)$ with low max divergence above. Surprisingly, while we are trying to compress a sequence generated from an i.i.d. distribution, our prescribed choice of $Q$ is not i.i.d.!

For simplicity, we restrict ourselves to the case of binary sequences, i.e., $\mathcal{X} = \{0, 1\}$. For this case, an i.i.d. distribution is characterized by a single parameter $p \in [0, 1]$, namely the probability of 1 in the common pmf. Designing a distribution $Q(x_1, ..., x_n)$ is the same as designing $Q(x_{i+1}|x_1, ..., x_i)$ for every $1 \leq i \leq n - 1$ and every $(x_1, ..., x_i) \in \mathcal{X}^i$. This latter question is that of probability estimation – Given that the outcomes of the first $i$ coin tosses are $x_1, ..., x_i$, what is the probability that the next coin toss will show 1. We might be tempted to use the fraction of heads in the coin tosses up to now as our estimate of the probability of heads, but then if we toss a coin once and it shows heads, we will find it hard to believe that the probability of heads is 1.

Another heuristic is the so-called “Bayesian heuristic” which simply assumes a “prior distribution” for the unknown parameter $p$. Since we are not sure about which value of $p$ was used to generate the sequence, it is heuristically appealing to assume that $p$ itself is random and is chosen uniformly over $[0, 1]$ (it was heuristically appealing to Laplace who wanted to estimate the probability of sunrise tomorrow!). With this choice of prior on the parameter $p$, the posterior estimate $Q(1|x_1, ..., x_i)$ is given by

$$Q(X_{i+1} = 1|x_1, ..., x_i) = \frac{\mathbb{P}(X_{i+1} = 1, X_j = x_j, 1 \leq j \leq i)}{\mathbb{P}(X_j = x_j, 1 \leq j \leq i)} = \frac{\int_0^1 p^{k+1}(1-p)^{n-k} \, dp}{\int_0^1 p^k(1-p)^{n-k} \, dp},$$

where $k$ denotes the number of 1s in the sequence $(x_1, ..., x_i)$. Note that by integration by parts, for $n \geq 1$

$$\int_0^1 p^n(1-p)^n \, dp = \frac{n}{m+1} \int_0^1 p^{m+1}(1-p)^{n-1} \, dp = \frac{1}{\binom{m+n}{m}} \int_0^1 p^{n+m} \, dp = \frac{1}{(m + n + 1)\binom{m+n}{m}}.$$
Therefore, 
\[ Q(X_{i+1} = 1|x_1, ..., x_i) = \frac{k + 1}{i + 2}. \]

This probability estimate can be viewed as the usual empirical frequency (type) estimator, except that instead of using the number of times \( N(a|x) \) each symbol has occurred, we add 1 to the number of occurrences and use \( N(a|x) + 1 \). Thus, upon observing \( x \in X^n \), our estimate for the probability of 1 is 
\[ \frac{N(1|x) + 1}{N(1|x) + 1 + N(0|x) + 1} = \frac{N(1|x) + 1}{n + 2}. \]

Such an estimator is called an add-1 estimator. Variants which add \( \alpha > 0 \) instead of 1 are also popular; they, too, have interpretations as posterior probabilities for appropriately chosen priors.

As an example consider the sequence 0010. The probability \( Q \) assigned by our method to this sequence is given by 
\[ Q(0010) = Q(0)Q(0|0)Q(1|00)Q(0|001) = \frac{1}{2} \cdot \frac{2}{3} \cdot \frac{2}{4} \cdot \frac{3}{5}. \]

Similarly, for the sequence 0100
\[ Q(0100) = Q(0)Q(1|0)Q(0|01)Q(0|010) = \frac{1}{2} \cdot \frac{2}{3} \cdot \frac{2}{4} \cdot \frac{3}{5}. \]

The above calculations are revealing. The probabilities under \( Q \) depend only on the number of 1s in the sequence and are given by (verify this!)
\[ Q(x) = \frac{k!(n-k)!}{(n+1)!} = \frac{1}{(n+1)\binom{n}{k}}. \]
where $k$ denotes the number of 1s in $x$. Also, for any pmf $P = \text{Bernoulli}(p)$,

$$P^n(x) = p^k(1 - p)^{n-k}.$$ 

Thus, denoting by $k_x$ the number of 1s in $x$,

$$D(P^n||Q) = \sum_x P^n(x) \log \frac{P^n(x)}{Q(x)}$$

$$= \sum_x P^n(x) \log(n + 1) \binom{n}{k_x} P^n(x)$$

$$= \log(n + 1) + \sum_x P^n(x) \log \binom{n}{k_x} - H(P^n).$$

Next, we note by Lemma 18.3 that (this is one of the HW problems)

$$\binom{n}{k} \leq 2^{nh(k/n)}.$$ 

Therefore,

$$D(P^n||Q) \leq \log(n + 1) + n \mathbb{E}_{P^n} \left[ h \left( \frac{k_{X^n}}{n} \right) \right] - nh(p)$$

$$\leq \log(n + 1),$$ 

where in the last step we have used the concavity of binary entropy function $h$ and the fact $\mathbb{E}_{P^n}[k_{X^n}] = np$.

Therefore, the proposed code which uses arithmetic coding together with the add-1 rule for probability assignment has regret less than $\log(n + 1)$. As discussed earlier, this is optimal up to constant factors. In fact, the optimal factor of $1/2$ can be attained using an add-1/2 rule for probability assignment. Note that the decoder for arithmetic code can estimate the probability of the next symbol to be decoded using the add-1 rule with the sequence decoded up to that point. Therefore, this scheme is completely online and the
sequence of symbols can be decoded in a FIFO manner in a single pass.