Electrical Engineering 229A Lecture 11 Notes

Daniel Raban

September 30, 2021

1 Shanon-Fano-Elias, Arithmetic, and Lempel-Ziv Coding

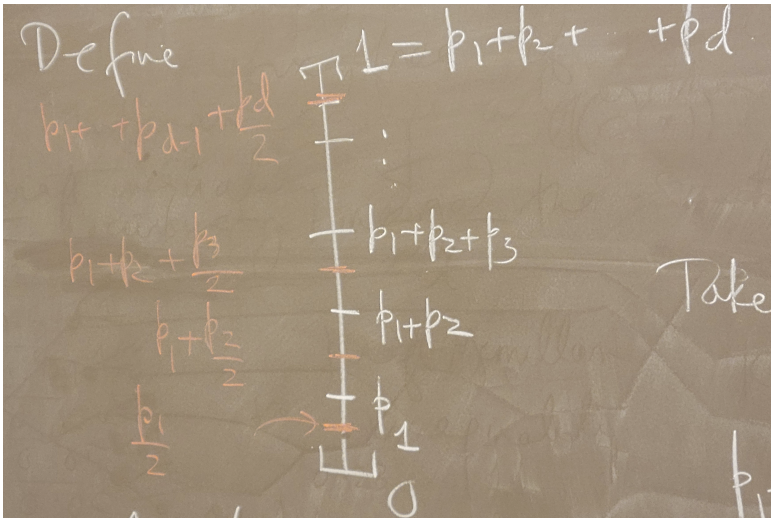
1.1 Shannon-Fano-Elias coding and arithmetic coding

Last time, we introduced the **Shannon-Fano-Elias coding** scheme. Suppose we have an alphabet $\mathscr{X}$ of size d, which we can assume are real numbers $a_1 < a_2 < \cdots < a_d$. If X is an $\mathscr{X}$-valued random variable, we can view it as a real-valued random variable. Then we have the Cumulative Distribution Function (CDF) of X:

$$F_X(x) = \mathbb{P}(X \leq x),$$

defined for all $x \in \mathbb{R}$.

Let $p_i := \mathbb{P}(X = a_i)$ for $i = 1, \ldots, d$, and label the values of the CDF on the interval $[0, 1]$ as follows.



Then label the midpoints of these values,

$$Q_1 = \frac{p_1}{2}, Q_2 = p_1 + \frac{p_2}{2}, \ldots, Q_d = \ldots, p_1 + \cdots + p_{d-1} + \frac{p_d}{2}.$$

Expand these in their binary representations, and truncate this representation to

$$\ell_i := \left\lceil \log \frac{1}{p_i} \right\rceil + 1$$

bits.

Example 1.1. Let $\mathscr{X} = \{1, 2, 3\}$ with $(p_1, p_2, p_3) = (1/4, 1/8, 5/8)$. Then the midpoints are

$$\frac{1}{8} = 0.001, \qquad \frac{3}{16} = 0.101, \qquad \frac{11}{16} = 0.1011.$$

Now we get the lengths

$$\ell_1 = \left\lceil \log \frac{1}{1/4} \right\rceil + 1 = 3,$$

$$\ell_2 = \left\lceil \log \frac{1}{1/8} \right\rceil + 1 = 3,$$

$$\ell_3 = \left\lceil \log \frac{1}{5/8} \right\rceil + 1 = 2.$$

So the Shannon-Fano-Elias code is

$$1 \mapsto 001, \qquad 2 \mapsto 010, \qquad 3 \mapsto 10.$$

Proposition 1.1. *This scheme gives a prefix-free code for* $(p_i, i \in \mathscr{X})$.

Proof. Think of the interval $[p_i + \cdots + p_{i-1}, p_1 + \cdots + p_i)$ as being *owned* by the symbol i, where the left endpoint is 0 for $i = 1$. Q_i is the midpoint of this interval. Let $(Q_i)_{\ell_i}$ denote the ℓ_i-truncation of the binary representation of Q_i. Observe that

$$Q_i - \sum_{j=1}^{i-1} p_j = \frac{p_i}{2} \geq \frac{1}{2^{\ell_i}}$$

because $\ell_i \geq \log \frac{1}{p_i} + 1$, so $2^{\ell_i - 1} \geq \frac{1}{p_i}$, which gives $\frac{p_i}{2} \geq \frac{1}{2^{\ell_i}}$.

Suppose that

$$(Q_i)_{\ell_i} = 0.b_1 b_2 \cdots b_{\ell_i}.$$

Consider

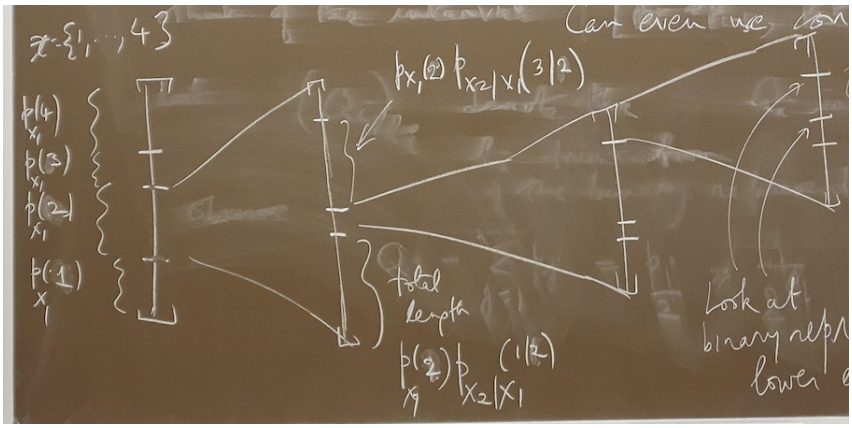
$$\left[0.b_1 b_2 \cdots b_{\ell_i}, 0.b_1 \cdots b_{\ell_i} + \frac{1}{2^{\ell_i}}\right).$$

The binary representations of the real numbers in this interval are the continuations of the binary tree of the string $b_1 \cdots b_{\ell_i}$, i.e. those that might violate the prefix-free condition. But the inequality $Q_i - \sum_{j=1}^{i-1} p_j \geq 1/2^{\ell_i}$ tells us that this interval falls inside the interval $[p_i + \cdots + p_{i-1}, p_1 + \cdots + p_i)$ owned by the symbol i. Hence, no $(Q_k)_{\ell_k}$ for $k \neq i$ can have $(Q_i)_{\ell_i}$ as a prefix. □

Observe that we also get

$$\sum_{i=1}^{d} p_i \left\lceil \log \frac{1}{p_i} \right\rceil \leq H(X) + 2.$$

We can now amortize the 2 over multiple symbols (say n symbols) in the naive way by implementing a scheme on $\mathscr{X}^n$ (instead of $\mathscr{X}$). More interesting is to break $[\sum_{j-1}^{i-1} p_j, \sum_{j=1}^{i} p_j)$ itself into d subintervals and so on, iteratively. We can even use conditional probabilities for this.



Look at the binary representations of the upper and lower ends, and spit out the prefix they agree on.

1.2 Lempel-Ziv coding and comma-free coding of natural numbers

Imagine you've seen an infinite number of symbols from an ergodic source, and you see a new symbol. How will you compress the new symbol? We want to represent a symbol in context of its past. This is leading up to the Lempel-Ziv coding scheme from 1977.

First, we need a "comma free" representation of natural numbers. Imagine a transmitter has a natural number and needs to send a bit string from which the receiver can figure out this integer. Here is a solution to this problem due to Elias.[1]

The integer n can be written as a bit string of length $\tau(n)$ (so $2^{\tau(n)} \geq n$). Then $\tau(n)$ can be written as a bit string of length $\phi(n)$ (so $2^{\phi(n)} \geq \tau(n)$). We could go on like this for a few stages, but for us this is good enough. To send $\phi(n)$, send

$$\underbrace{000\cdots0}_{\phi(n) \text{ zeroes}}\, 1.$$

[1] There are many solutions. You can try to figure one out for yourself.

The number of bits used in this scheme is

$$\begin{aligned} 2\phi(n) + 1 + \tau(n) &\leq \lceil \log n \rceil + 2\lceil \log \lceil \log n \rceil \rceil + 1 \\ &\leq \log n + 2 \log \log n + 5. \end{aligned}$$

The key idea in LZ'77 is to compress new samples in the context of the infinite past by transmitting how far back in the past to look to see the current samples in the context of its own past. This works for a general stationary, ergodic process, but to understand why this works, we will consider the iid case, working with blocks. Suppose we have

$$(\dots, x_{-2}, x_{-1}, x_0, x_1, \dots, x_{k-1}, \dots),$$

where $x_{\leq -1}$ is to be shared with the receiver, and $x_0, x_1, \dots, x_{k-1}$ is to be conveyed. Find

$$\inf\{L \geq l : (x_0, \dots, x_{k-1}) = (x_{-L} x_{-L+1}, \dots, x_{-L+k-1})\}.$$

Let $p(a_0^{k-1}) := \mathbb{P}((X_0, \dots, X_{k-1}) = a_0^{k-1})$. We can show that the comma–free encoding needs $\leq (1+\varepsilon) H(X_1)$ bits. We will do this next time.