Indian Institute of Technology Bombay

Department of Electrical Engineering

Handout 5 EE 708 Information Theory and Coding
Tutorial 2 Feb 9, 2018

Q1: Arithmetic and Shannon-Fano-Elias Codes: Arithmetic coding is a widely used data compression technique which has many advantages[1].

Remark: Arithmetic Coding operates on sequences of source symbols or strings. While mapping strings to values in the encoding process, let us be consistent with the lexicographical ordering of strings. In particular the ordering is captured by the following notation,

$$u_1 u_2 \cdot \cdot u_n > v_1 v_2 \cdot \cdot v_n$$

if $u_i > v_i$, where i is the first position that they differ.

Arithmetic codes are related to the so called Shannon-Fano-Elias coding, and their *cousin*, **Elias** codes. We start with a seemingly unrelated question on continuous valued random variables.

a) Let random variable X admit a pdf $f_X(x)$, and is supported on a connected interval, i.e. we can define an inverse for the CDF $F_X(x)$. Suppose we independently draw a continuous valued random variable $S \sim f_X(\cdot)$ and take

$$\mathbf{Y} = F_X(\mathbf{S}).$$

where we used bold-face to denote that expression is a mapping of random variables. Draw $F_Y(y)$ in the figure below (say with dashed lines).

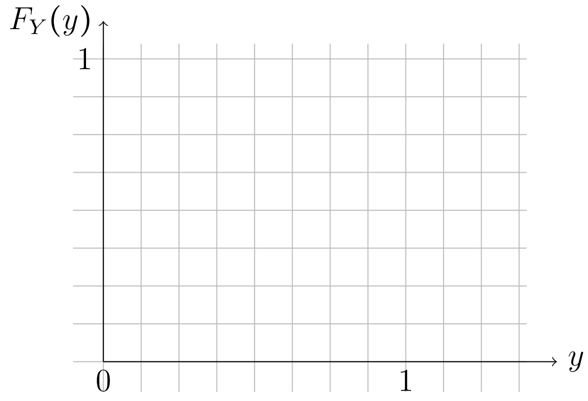


Figure 1: Fig. for (a)-(b)

(b) Let X be a discrete random variable in $0, 1, 2, 3$ with probabilities q_1, q_2, q_3, q_4 respectively, having CDF $F_X(\cdot)$. With $\mathbf{Y} = F_X(\mathbf{X})$, draw $F_Y(y)$ in Figure 1. Assume right continuity for all the calculations.

(c) Show that

$$H(Y) = H(X)$$

(d) Let $v_i = \sum_{j<i} q_j + \frac{q_i}{2}$ represent the mid-points of the jumps in the CDF. Suppose we quantize the binary representation of v_i to l_i bits. By taking $l_i = \lceil \log \frac{1}{q_i} \rceil + 1$, construct a prefix free code for encoding this source. Will such prefix free codes exist for a general discrete random variables X taking finite values, for the same choice of lengths as above.

[1]parts of the practical implementations are proprietary

(e) Assume that $q_1 = p, q_2 = 1 - p$, $p > \frac{1}{2}$. Draw $F_Y(y)$ in Figure 2.

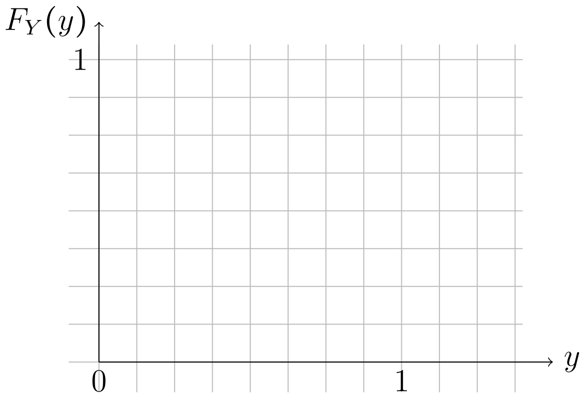
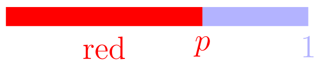


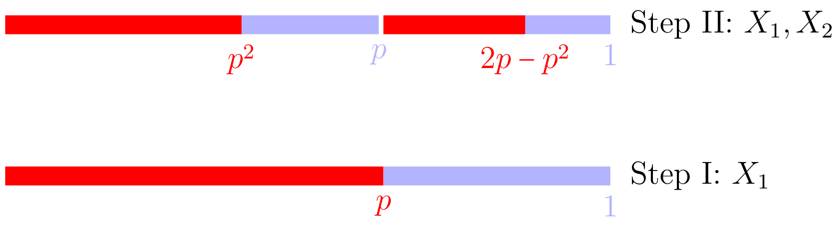
Figure 2: Fig for part (e)-(f)

(f) Consider $\bar{X} = X_1, \cdot\cdot, X_k$ generated iid, and $Y = F_{\bar{X}}(X_1, \cdot\cdot, X_k)$. Can you draw $F_Y(y)$ on Figure 2 for $k = 2, 3, 4$.

(g) Read and understand Section Consider a uniform random variable U in $[0, 1]$. Let us partition the unit interval into 2 parts, of lengths p and $1 - p$.



Suppose we are required to convey whether red occurred or not, based on U. This is equivalent to conveying a Bernoulli RV X with $P(X = 1) = p$. We can generalize this picture for $k = 2$ and more.



Like part (f), conveying $\bar{X}$ is equivalent to identifying one among the four sub-segments. Equivalently, we can toss a uniform U and the occurred segment above needs to convey its identity. See that there is an one to one map between any $X_1, \cdot\cdot, X_k$ and the mid-point of $F_Y(y)$ based on our ordering of sequences (as in part d).

Let us look at the cumulative distribution function $F_{X^n}(x^n)$ sequentially in n. Then, in an equivalent sense, we will convey the identity of the line segment which contains $F_{X^n}(x^n)$ for the occurred sequence x^n. To this end, we will progressively localize the line segment in a dyadic fashion. Assume that the location is known to be in $[a, b]$. At every step, we will try to localize the line segment to be on either side $\frac{a+b}{2}$. For example at the start $a = 0, b = 1$ and we will check the line-segment to be on either side of $\frac{1}{2}$.

At $n = 1$, if $x_1 = 1$, we will send(output) 1, since the line segment $(p, 1)$ is to the right of $\frac{1}{2}$. So the new locality is $[a, b] = [\frac{1}{2}, 1]$. Can x_1 be further localized to sub-halves inside $[a, b]$, then we will output the corresponding bit and again update the interval $[a, b]$ in an iterative fashion. At some point, we will not be able to localize to either side of $\frac{a+b}{2}$ inside $[a, b]$. In this case, we will take the next source symbol, say x_2 and consider $F_{X_1,X_2}(x_1, x_2)$. The corresponding line segment will be contained in the last known locality $[a, b]$. We will now localize $F_{X_1,X_2}(x_1, x_2)$ within the interval $[a, b]$. The general procedure is, with symbol x_i localized to be in $[a, b]$, Elias's Algorithm, $Algo_E$

1. Every time we localize further
   - to the left, output the bit 0, update $a = a, b = \frac{a+b}{2}$.
   - to the right, output the bit 1, update $a = \frac{a+b}{2}, b = b$.
2. If no further localization,
   - Output nothing, set $a = a, b = b$.
   - Take the next symbol, say x_{i+1} and consider the line segment corresponding to $x_1, x_2, \cdot\cdot, x_i, x_{i+1}$ and go back to the start.

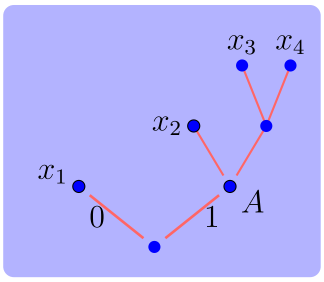
For example, if $x_1 = 0$, then no output is produced as we cannot localize the segment to either side. In this case, we will look at x_2 and try to localize $F_{X_1,X_2}(x_1, x_2)$. Notice that the segments get smaller and smaller as more symbols come in.
(h) Using the above procedure, find the Elias code for every binary sequences x_1, x_2, x_3, assuming the source is binary iid with $p = 0.74$. Also argue that the code is not prefix-free and not even uniquely decodable.

(i) How can you convert this to a prefix-free code. Do you now understand how compression is happening in Shannon-Fano-Elias and hence Arithmetic coding.

(j) A big advantage of this scheme is that it is **sequential**. Explain what you understand about the previous statement.

FYI: A finite precision implementation of this scheme is what is known as Arithmetic Coding. What we described was developed by Elias.

Q2) Tunstall Coding The data compression codes that we studied in the classroom converted a fixed number of input symbols to variable length output codewords. The current question aims to convey a dual approach: take a variable number of input symbols and convert it into fixed-length output sequences.
(a) Consider a directed code tree of a given variable length prefix free code (say Huffman code). Let us call the leaves of the tree as the **terminal vertexes** or **terminal nodes**. We can name the terminal nodes by the corresponding input symbol/sequence.



Consider any intermediate node (vertex), say node A. Let $B_j^A, 1 \le j \le q$ be the set of terminal vertexes which can be reached from A in j steps. In other words, from A, there exists a sequence of child branches leading to each terminal node $B_i^A, 1 \le i \le q$. Define,

$$P(A) \triangleq \sum_i p(B_i^A).$$

The above figure illustrates the idea. Here x_1, x_2, x_3 and x_4 are the terminal nodes. For the node marked A there, $B_1^A = x_2, B_2^A = \{x_3, x_4\}$.
Show that for the code-tree of any prefix-free code

$$\sum_A P(A) = L_{avg}$$

where the summation is over all the nodes **excluding** the terminal nodes, and L_{avg} is the average code-length.

(b) Let $X \in \{A, B, C\}$ be some source with probability $p_A = 0.7$, $p_B = 0.2$, $p_C = 0.1$. Assume a long sequence of iid realizations of X. We are also free to append up to 2 dummy inputs at the end of the sequence. This is just to sensibly terminate our coding scheme, and can be neglected for computations. Now consider the following coding scheme, which maps the occurrence of symbols shown on the left to the corresponding code bits on the right.

$$\begin{array}{ll} AAA \to 000 & AB \to 011 \\ AAB \to 001 & AC \to 100 \\ AAC \to 010 & B \to 101 \\ & C \to 110 \end{array}$$

Is this code **uniquely encodable**?. i.e., is there a **unique** code sequence corresponding to every input sequence?. Explain how this code achieves compression over simply using 2 bits to convey A, B, C.

(c) Notice that in part (b), variable length input streams (or **words**) are mapped to a fixed length codewords. Let us call this C_3 scheme, as the codewords are 3 bit sequences. In general we will say C_k **scheme** for any uniquely encodable strategy with output codewords of k bits. With no loss of generality we can strike off any unused codewords (like 111 above) and consider the minimal set.
Show that the code in part (b) is the best C_3 scheme in terms of compression achieved (i.e. $\frac{k}{N_k}$) for the source X in part (b).

(d) Tunstall Code: It turns out that the best C_k coding scheme can be constructed in an iterative fashion. Given a set of **words** and their probabilities, pick the highest probable word at any stage and append all possible source symbols to it. As an example, for the source we consider, from the initial set $S_1 = \{A, B, C\}$ to $S_2 = \{AA, AB, AC, B, C\}$ and $S_3 = \{AAA, AAB, AAC, AB, AC, B, C\}$. Iteratively, find the highest probability element of S_j and append each possible source alphabet to it to get S_{j+1}. Continue this if $|S_{j+1}| \le 2^k$, otherwise terminate with the best C_k scheme.

Show that

$$H(S_3) = N_3.H(X)$$

where N_k is the average number of source symbols getting encoded in C_k scheme.
Hint: $H_m(p_1, p_2, \cdots, p_m) = H_{m-1}(p_1 + p_2, p_3, \cdots, p_m) + (p_1 + p_2)H_b(\frac{p_1}{p_1+p_2})$, and use part (a).

(e) *(Bonus)* We can show that for the C_k scheme using Tunstall coding, and large k

$$H(X) \le \frac{k}{N_k} \le H(X) + 1$$

(While a proof will be presented in class, there is scope for an innovative but elementary technique, please attempt). Compare the efficiency of Huffman and Tunstall codes.