

**Problem Set 3 solutions**

November 2, 2007

5.3 *Slackness in the Kraft inequality.* Instantaneous codes are prefix free codes, i.e., no codeword is a prefix of any other codeword. Let  $l_{max} = \max\{l_1, l_2, \dots, l_m\}$ . There are  $D^{l_{max}}$  sequences of length  $l_{max}$ . Of these sequences,  $D^{l_{max}-l_i}$  start with the  $i$ -th codeword. Because of the prefix condition, any sequence of length  $D^{l_{max}}$  whose prefix is the  $i$ -th codeword must be distinct from a sequence of length  $D^{l_{max}}$  whose prefix is the  $j$ -th codeword, for any  $i \neq j$ . Hence the total number of sequences which start with some codeword is  $\sum_{i=1}^m D^{l_{max}-l_i} = D^{l_{max}} \sum_{i=1}^m D^{-l_i} < D^{l_{max}}$ . Hence there are sequences which do not start with any codeword. These and all longer sequences with these length  $l_{max}$  sequences as prefixes cannot be decoded. (This situation can be visualized with the aid of a tree.)

Alternatively, we can map codewords onto dyadic intervals on the real line corresponding to real numbers whose decimal expansions start with that codeword. Since the length of the interval for a codeword of length  $l_i$  is  $D^{-l_i}$ , and  $\sum D^{-l_i} < 1$ , there exists some interval(s) not used by any codeword. The binary sequences in these intervals do not begin with any codeword and hence cannot be decoded.

5.5 *More Huffman codes.* The Huffman code for the source with probabilities  $(\frac{1}{3}, \frac{1}{5}, \frac{1}{5}, \frac{2}{15}, \frac{2}{15})$  has codewords  $\{00, 10, 11, 010, 011\}$ .

To show that this code (\*) is also optimal for  $(1/5, 1/5, 1/5, 1/5, 1/5)$  we have to show that it has minimum expected length, that is, no shorter code can be constructed without violating  $H(X) \leq EL$ .

$$H(X) = \log 5 = 2.32 \text{ bits.} \quad (1)$$

$$E(L(*)) = 2 \times \frac{3}{5} + 3 \times \frac{2}{5} = \frac{12}{5} \text{ bits.} \quad (2)$$

Since

$$E(L(\text{any code})) = \sum_{i=1}^5 \frac{l_i}{5} = \frac{k}{5} \text{ bits} \quad (3)$$

for some integer  $k$ , the next lowest possible value of  $E(L)$  is  $11/5 = 2.2 \text{ bits} < 2.32 \text{ bits}$ . Hence (\*) is optimal.

Note that one could also prove the optimality of (\*) by showing that the Huffman code for the  $(1/5, 1/5, 1/5, 1/5, 1/5)$  source has average length  $12/5$  bits. (Since each Huffman code produced by the Huffman encoding algorithm is optimal, they all have the same average length.)

5.9 *Optimal code lengths that require one bit above entropy.* There is a trivial example that requires almost 1 bit above its entropy. Let  $X$  be a binary random variable with probability of  $X = 1$  close to 1. Then entropy of  $X$  is close to 0, but the length of its optimal code is 1 bit, which is almost 1 bit above its entropy.

5.12 *Shannon codes and Huffman codes.*

- (a) Applying the Huffman algorithm gives us the following table

Code	Symbol	Probability			
0	1	1/3	1/3	2/3	1
11	2	1/3	1/3	1/3	
101	3	1/4	1/3		
100	4	1/12			

which gives codeword lengths of 1,2,3,3 for the different codewords.

- (b) Both set of lengths 1,2,3,3 and 2,2,2,2 satisfy the Kraft inequality, and they both achieve the same expected length (2 bits) for the above distribution. Therefore they are both optimal.
- (c) The symbol with probability 1/4 has an Huffman code of length 3, which is greater than  $\lceil \log \frac{1}{p} \rceil$ . Thus the Huffman code for a particular symbol may be longer than the Shannon code for that symbol. But on the average, the Huffman code cannot be longer than the Shannon code.

#### 5.24 Optimal codes for uniform distributions.

- (a) For uniformly probable codewords, there exists an optimal binary variable length prefix code such that the longest and shortest codewords differ by at most one bit. If two codes differ by 2 bits or more, call  $m_s$  the message with the shorter codeword  $C_s$  and  $m_\ell$  the message with the longer codeword  $C_\ell$ . Change the codewords for these two messages so that the new codeword  $C'_s$  is the old  $C_s$  with a zero appended ( $C'_s = C_s0$ ) and  $C'_\ell$  is the old  $C_\ell$  with a one appended ( $C'_\ell = C_\ell1$ ).  $C'_s$  and  $C'_\ell$  are legitimate codewords since no other codeword contained  $C_s$  as a prefix (by definition of a prefix code), so obviously no other codeword could contain  $C'_s$  or  $C'_\ell$  as a prefix. The length of the codeword for  $m_s$  increases by 1 and the length of the codeword for  $m_\ell$  decreases by at least 1. Since these messages are equally likely,  $L' \leq L$ . By this method we can transform any optimal code into a code in which the length of the shortest and longest codewords differ by at most one bit. (In fact, it is easy to see that every optimal code has this property.)

For a source with  $n$  messages,  $\ell(m_s) = \lfloor \log_2 n \rfloor$  and  $\ell(m_\ell) = \lceil \log_2 n \rceil$ . Let  $d$  be the difference between  $n$  and the next smaller power of 2:

$$d = n - 2^{\lfloor \log_2 n \rfloor}.$$

Then the optimal code has  $2d$  codewords of length  $\lceil \log_2 n \rceil$  and  $n - 2d$  codewords of length  $\lfloor \log_2 n \rfloor$ . This gives

$$\begin{aligned} L &= \frac{1}{n} (2d \lceil \log_2 n \rceil + (n - 2d) \lfloor \log_2 n \rfloor) \\ &= \frac{1}{n} (n \lfloor \log_2 n \rfloor + 2d) \\ &= \lfloor \log_2 n \rfloor + \frac{2d}{n}. \end{aligned}$$

Note that  $d = 0$  is a special case in the above equation.

- (b) The average codeword length equals entropy if and only if  $n$  is a power of 2. To see this, consider the following calculation of  $L$ :

$$L = \sum_i p_i \ell_i = - \sum_i p_i \log_2 2^{-\ell_i} = H + D(p||q),$$

where  $q_i = 2^{-\ell_i}$ . Therefore  $L = H$  only if  $p_i = q_i$ , that is, when all codewords have equal length, or when  $d = 0$ .

- (c) For  $n = 2^m + d$ , the redundancy  $r = L - H$  is given by

$$\begin{aligned} r &= L - \log_2 n \\ &= \lfloor \log_2 n \rfloor + \frac{2d}{n} - \log_2 n \\ &= m + \frac{2d}{n} - \log_2(2^m + d) \\ &= m + \frac{2d}{2^m + d} - \frac{\ln(2^m + d)}{\ln 2}. \end{aligned}$$

Therefore

$$\frac{\partial r}{\partial d} = \frac{(2^m + d)(2) - 2d}{(2^m + d)^2} - \frac{1}{\ln 2} \cdot \frac{1}{2^m + d}$$

Setting this equal to zero implies  $d^* = 2^m(2 \ln 2 - 1)$ . Since there is only one maximum, and since the function is convex  $\cap$ , the maximizing  $d$  is one of the two integers nearest  $(.3862)(2^m)$ . The corresponding maximum redundancy is

$$\begin{aligned} r^* &\approx m + \frac{2d^*}{2^m + d^*} - \frac{\ln(2^m + d^*)}{\ln 2} \\ &= m + \frac{2(.3862)(2^m)}{2^m + (.3862)(2^m)} - \frac{\ln(2^m + (.3862)2^m)}{\ln 2} \\ &= .0861. \end{aligned}$$

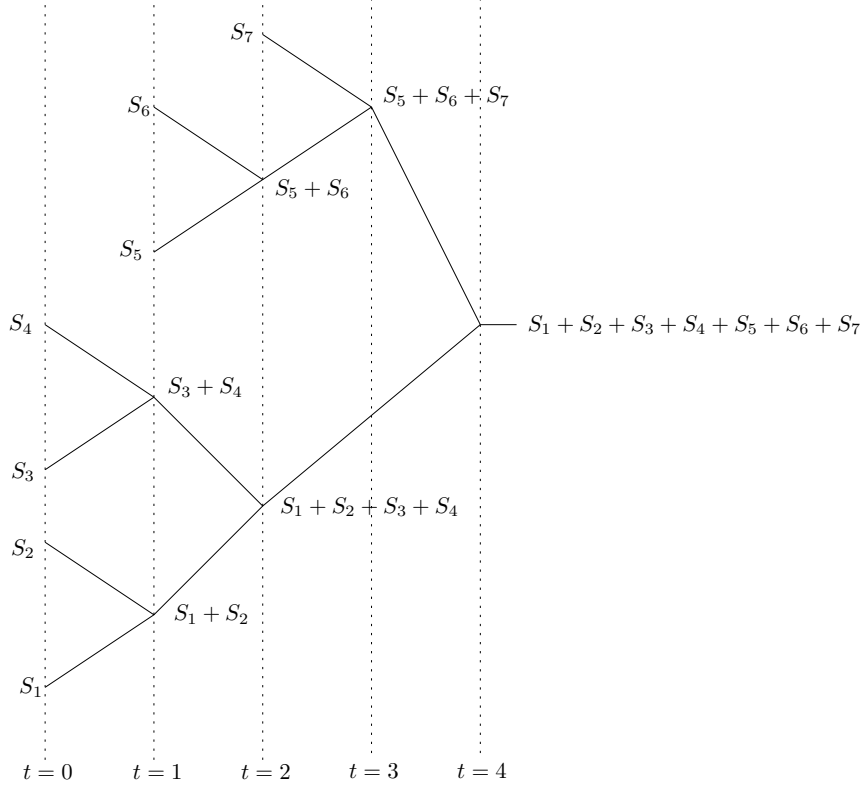
This is achieved with arbitrary accuracy as  $n \rightarrow \infty$ . (The quantity  $\sigma = 0.0861$  is one of the lesser fundamental constants of the universe. See Robert Gallager[1].)

### 5.34 Huffman algorithm for tree construction.

First, without loss of generality, in the following we will assume  $T_1 = 0$ . In particular, we can define a new set of times,  $T'_i = T_i - T_1, \forall i$ . Clearly, the optimal algorithm is the same for both.

Now, consider an example of the Huffman tree construction algorithm, where  $T_1 = 0, T_2 = 0, T_3 = 0, T_4 = 0, T_5 = 1, T_6 = 1, T_7 = 2$ . The following figure illustrates one possible solution using the Huffman algorithm. Note that in general, just as in Huffman codes, there may be more than one possible tree, but all Huffman trees will correspond to the same overall computation time.

- (a) To compute the sum of  $n$  binary numbers, we need to perform exactly  $n - 1$  binary additions. Furthermore, we should combine any available summands (i.e., terms of the form  $S_i, S_j + S_k$ , etc.) as soon as they are available. In other words, we can't decrease the



overall time required to perform all  $n - 1$  additions by purposely delaying the addition of any available summands.

More formally, suppose that two algorithms are identical up to time  $t = t_0$ . At time  $t_0$ , suppose that there are  $K$  available summands (in our running example, for  $t = 1$  we have  $K = 4$ , specifically the summands are  $S_1 + S_2$ ,  $S_3 + S_4$ ,  $S_5$  and  $S_6$ ).

Algorithm 1 (Greedy): We perform  $j = \lfloor \frac{K}{2} \rfloor$  additions, using  $j$  two-input gates in parallel, and  $K = 2 * j + m$ .

Algorithm 2 : Suppose we perform  $j' < j$  additions, where  $j = j' + i, i > 0$ , then  $K = 2 * j' + m'$ , and it is easy to see that  $m' = m + 2i$ .

Now, at time  $t = t_0 + 1$ , suppose  $B$  \*new\* binary input signals (i.e.,  $S_k$  such that  $T_k = t_0 + 1$ ) become available. In our running example, at  $t = 2$  we have  $S_7$  becoming available, so  $B = 1$  at  $t = 2$ .

Algorithm 1: We have  $j + m + B$  summands available, and thus the greedy algorithm computes  $\lfloor \frac{j+m+B}{2} \rfloor$  additions, and thus it has computed a total of  $j + \lfloor \frac{j+m+B}{2} \rfloor$  additions in the last two stages.

Algorithm 2: We have  $j' + m' + B = j' + m + 2i + B$  summands, and we could compute as many as  $\lfloor \frac{j'+m+2i+B}{2} \rfloor = i + \lfloor \frac{j'+m+B}{2} \rfloor$  additions in this stage, for a total of  $j' + i + \lfloor \frac{j'+m+B}{2} \rfloor = j + \lfloor \frac{j'+m+B}{2} \rfloor$  additions in the last two stages.

Clearly, since  $j' < j$ , we have that

$$j + \lfloor \frac{j + m + B}{2} \rfloor \geq j + \lfloor \frac{j' + m + B}{2} \rfloor,$$

and thus the greedy algorithm computes at least as many additions as the algorithm that leaves some additions ‘for later’. Furthermore, continuing inductively with this line of reasoning, starting from  $t_0 = T_2$ , we can argue that the greedy algorithm always does at least as well as any algorithm that saves work for later.

- (b) From its definition,  $C(T)$  is the total length of time required to compute the desired sum, and we know that the Huffman algorithm minimizes this length of time, and thus it minimizes  $C(T)$ .
- (c) This is similar to the Kraft Inequality. First, every  $S_i$  must be represented by a leaf node on the tree. Furthermore, for a complete binary tree of depth  $C(T)$ , there are  $2^{C(T)}$  leaf nodes. Now, since the leaf node corresponding to  $S_i$  in the Huffman tree cannot occur prior to  $t = T_i$  (that is, we can’t add it to another value until it is available to us, and it doesn’t become available to us until  $T_i$ ), then at least  $2^{T_i}$  leaf nodes in the complete binary tree must be children of the node corresponding to  $S_i$ . Furthermore, since each  $S_i$  is assigned to a unique leaf node, none of their children (again, in the corresponding full binary tree) can overlap. Therefore,

$$\sum_i 2^{T_i} \leq 2^{C(T)}$$

and thus

$$C(T) \geq \log_2 \left( \sum_i 2^{T_i} \right).$$

- (d) We will illustrate the existence of such a tree by construction. First, let  $n = \lceil \log_2 \left( \sum_i 2^{T_i} \right) \rceil$ . Then, we have

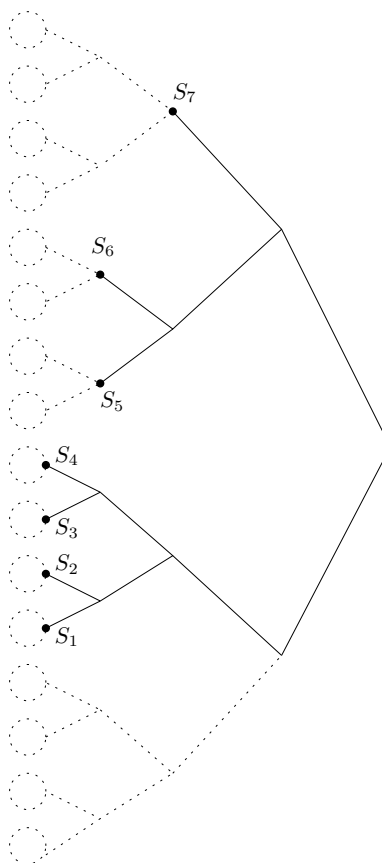
$$n = \left\lceil \log_2 \left( \sum_i 2^{T_i} \right) \right\rceil < \log_2 \left( \sum_i 2^{T_i} \right) + 1 = \log_2 \left( \sum_i 2^{T_i+1} \right).$$

Next, we draw the complete binary tree of depth  $n$ , and label the  $2^n$  leaf nodes from 0 to  $2^n - 1$ . We then assign the first  $(2^n - \sum_i 2^{T_i})$  leaf nodes to be dummy nodes. Next, we assign the next  $2^{T_1}$  available leaf nodes to input  $S_1$ , and similarly the following  $2^{T_2}$  available leaf nodes to input  $S_2$ , and so on, until we assign the last  $2^{T_m}$  available leaf nodes to input  $S_m$ .

Now, by trimming and appropriately assigning tree nodes to the  $S_i$ , we can transform this tree into the desired form. First, we remove any nodes (and their descendants) if all of their descendants are dummy leaf nodes. Similarly, if a node has both dummy and non-dummy descendants, we keep the node, but remove the subtree corresponding to the dummy descendants. Now, for input  $S_i$ , where  $T_i = k$ , we assign the unique node at level  $k$  whose  $2^k$  children are exactly those leaf nodes that were originally assigned to input  $S_i$ , from the complete binary tree. In particular, we label this node  $S_i$ , and remove its descendants. Repeating this process for all inputs, we arrive at a valid addition tree. Furthermore, since  $C(T)$  is the depth of this pruned tree, and since input  $S_1$  occurs at  $T_1 = 0$ , we have that  $C(T) = n$  and

$$C(T) < \log_2 \left( \sum_i 2^{T_i} \right) + 1.$$

To illustrate the preceding construction algorithm, we perform it for the same set of  $T_i$  as in our running examples. In the figure below, the dashed portions of the tree represent trimmed branches. Note that this tree is identical to that found earlier by the Huffman algorithm. Indeed, the preceding construction algorithm always finds a tree that minimizes the computation time, and thus performs as well as the Huffman algorithm.



#### 4.1 Doubly Stochastic Matrices.

(a)

$$H(\mathbf{b}) - H(\mathbf{a}) = -\sum_j b_j \log b_j + \sum_i a_i \log a_i \quad (4)$$

$$= \sum_j \sum_i a_i P_{ij} \log \left( \sum_k a_k P_{kj} \right) + \sum_i a_i \log a_i \quad (5)$$

$$= \sum_i \sum_j a_i P_{ij} \log \frac{a_i}{\sum_k a_k P_{kj}} \quad (6)$$

$$\geq \left( \sum_{i,j} a_i P_{ij} \right) \log \frac{\sum_{i,j} a_i}{\sum_{i,j} b_j} \quad (7)$$

$$= 1 \log \frac{m}{m} \tag{8}$$

$$= 0, \tag{9}$$

where the inequality follows from the log sum inequality.

(b) If the matrix is doubly stochastic, the substituting  $\mu_i = \frac{1}{m}$ , we can easily check that it satisfies  $\mu = \mu P$ .

(c) If the uniform is a stationary distribution, then

$$\frac{1}{m} = \mu_i = \sum_j \mu_j P_{ji} = \frac{1}{m} \sum_j P_{ji}, \tag{10}$$

or  $\sum_j P_{ji} = 1$  or that the matrix is doubly stochastic.

#### 4.7 Entropy rates of Markov chains.

(a) The stationary distribution is easily calculated. (See EIT pp. 62–63.)

$$\mu_0 = \frac{p_{10}}{p_{01} + p_{10}}, \quad \mu_1 = \frac{p_{01}}{p_{01} + p_{10}}.$$

Therefore the entropy rate is

$$H(X_2|X_1) = \mu_0 \mathcal{H}(p_{01}) + \mu_1 \mathcal{H}(p_{10}) = \frac{p_{10} \mathcal{H}(p_{01}) + p_{01} \mathcal{H}(p_{10})}{p_{01} + p_{10}}.$$

(b) The entropy rate is at most 1 bit because the process has only two states. This rate can be achieved if (and only if)  $p_{01} = p_{10} = 1/2$ , in which case the process is actually i.i.d. with  $\Pr(X_i = 0) = \Pr(X_i = 1) = 1/2$ .

(c) As a special case of the general two-state Markov chain, the entropy rate is

$$H(X_2|X_1) = \mu_0 \mathcal{H}(p) + \mu_1 \mathcal{H}(1) = \frac{\mathcal{H}(p)}{p + 1}.$$

(d) By straightforward calculus, we find that the maximum value of  $H(X)$  of part (c) occurs for  $p = (3 - \sqrt{5})/2 = 0.382$ . The maximum value is

$$\mathcal{H}(p) = \mathcal{H}(1 - p) = \mathcal{H}\left(\frac{\sqrt{5} - 1}{2}\right) = 0.694 \text{ bits}.$$

Note that  $(\sqrt{5} - 1)/2 = 0.618$  is (the reciprocal of) the Golden Ratio.

(e) The Markov chain of part (c) forbids consecutive ones. Consider any allowable sequence of symbols of length  $t$ . If the first symbol is 1, then the next symbol must be 0; the remaining  $N(t - 2)$  symbols can form any allowable sequence. If the first symbol is 0, then the remaining  $N(t - 1)$  symbols can be any allowable sequence. So the number of allowable sequences of length  $t$  satisfies the recurrence

$$N(t) = N(t - 1) + N(t - 2) \quad N(1) = 2, N(2) = 3$$

(The initial conditions are obtained by observing that for  $t = 2$  only the sequence 11 is not allowed. We could also choose  $N(0) = 1$  as an initial condition, since there is exactly one allowable sequence of length 0, namely, the empty sequence.)

The sequence  $N(t)$  grows exponentially, that is,  $N(t) \approx c\lambda^t$ , where  $\lambda$  is the maximum magnitude solution of the characteristic equation

$$1 = z^{-1} + z^{-2}.$$

Solving the characteristic equation yields  $\lambda = (1 + \sqrt{5})/2$ , the Golden Ratio. (The sequence  $\{N(t)\}$  is the sequence of Fibonacci numbers.) Therefore

$$H_0 = \lim_{n \rightarrow \infty} \frac{1}{n} \log N(n) = \log(1 + \sqrt{5})/2 = 0.694 \text{ bits}.$$

Since there are only  $N(t)$  possible outcomes for  $X_1, \dots, X_t$ , an upper bound on  $H(X_1, \dots, X_t)$  is  $\log N(t)$ , and so the entropy rate of the Markov chain of part (c) is at most  $H_0$ . In fact, we saw in part (d) that this upper bound can be achieved.

#### 4.12 The entropy rate of a dog looking for a bone.

(a) By the chain rule,

$$\begin{aligned} H(X_0, X_1, \dots, X_n) &= \sum_{i=0}^n H(X_i | X^{i-1}) \\ &= H(X_0) + H(X_1 | X_0) + \sum_{i=2}^n H(X_i | X_{i-1}, X_{i-2}), \end{aligned}$$

since, for  $i > 1$ , the next position depends only on the previous two (i.e., the dog's walk is 2nd order Markov, if the dog's position is the state). Since  $X_0 = 0$  deterministically,  $H(X_0) = 0$  and since the first step is equally likely to be positive or negative,  $H(X_1 | X_0) = 1$ . Furthermore for  $i > 1$ ,

$$H(X_i | X_{i-1}, X_{i-2}) = H(.1, .9).$$

Therefore,

$$H(X_0, X_1, \dots, X_n) = 1 + (n - 1)H(.1, .9).$$

(b) From a),

$$\begin{aligned} \frac{H(X_0, X_1, \dots, X_n)}{n + 1} &= \frac{1 + (n - 1)H(.1, .9)}{n + 1} \\ &\rightarrow H(.1, .9). \end{aligned}$$

(c) The dog must take at least one step to establish the direction of travel from which it ultimately reverses. Letting  $S$  be the number of steps taken between reversals, we have

$$\begin{aligned} E(S) &= \sum_{s=1}^{\infty} s(.9)^{s-1}(.1) \\ &= 10. \end{aligned}$$

Starting at time 0, the expected number of steps to the first reversal is 11.

### 4.33 Chain Inequality.

First, Markovity implies that

$$I(X_2; X_3|X_4) \geq I(X_1; X_3|X_4). \quad (11)$$

In fact, the preceding can be derived by expanding  $I(X_1, X_2; X_3|X_4)$  in two different ways, as in the data-processing inequality.

Now,

$$I(X_2; X_3|X_4) = H(X_2|X_4) - H(X_2|X_3, X_4) = H(X_2|X_4) - H(X_2|X_3),$$

where the second equality follows by Markovity, and similarly

$$I(X_1; X_3|X_4) = H(X_1|X_4) - H(X_1|X_3).$$

Finally,

$$I(X_2; X_3|X_4) = H(X_2|X_4) - H(X_2|X_3) = I(X_2; X_3) - I(X_2; X_4)$$

and likewise

$$I(X_1; X_3|X_4) = H(X_1|X_4) - H(X_1|X_3) = I(X_1; X_3) - I(X_1; X_4).$$

Finally, substituting these into (11), we have

$$I(X_2; X_3) + I(X_1; X_4) \geq I(X_1; X_3) + I(X_2; X_4)$$

as desired.

### 7.1 Preprocessing the output.

- (a) The statistician calculates  $\tilde{Y} = g(Y)$ . Since  $X \rightarrow Y \rightarrow \tilde{Y}$  forms a Markov chain, we can apply the data processing inequality. Hence for every distribution on  $x$ ,

$$I(X; Y) \geq I(X; \tilde{Y}). \quad (12)$$

Let  $\tilde{p}(x)$  be the distribution on  $x$  that maximizes  $I(X; \tilde{Y})$ . Then

$$C = \max_{p(x)} I(X; Y) \geq I(X; Y)_{p(x)=\tilde{p}(x)} \geq I(X; \tilde{Y})_{p(x)=\tilde{p}(x)} = \max_{p(x)} I(X; \tilde{Y}) = \tilde{C}. \quad (13)$$

Thus, the statistician is wrong and processing the output does not increase capacity.

- (b) We have equality (no decrease in capacity) in the above sequence of inequalities only if we have equality in data processing inequality, i.e., for the distribution that maximizes  $I(X; \tilde{Y})$ , we have  $X \rightarrow \tilde{Y} \rightarrow Y$  forming a Markov chain.

### 7.3 Channels with memory have a higher capacity.

$$Y_i = X_i \oplus Z_i, \quad (14)$$

where

$$Z_i = \begin{cases} 1 & \text{with probability } p \\ 0 & \text{with probability } 1 - p \end{cases} \quad (15)$$

and  $Z_i$  are not independent.

$$\begin{aligned} I(X_1, X_2, \dots, X_n; Y_1, Y_2, \dots, Y_n) &= H(X_1, X_2, \dots, X_n) - H(X_1, X_2, \dots, X_n | Y_1, Y_2, \dots, Y_n) \\ &= H(X_1, X_2, \dots, X_n) - H(Z_1, Z_2, \dots, Z_n | Y_1, Y_2, \dots, Y_n) \\ &\geq H(X_1, X_2, \dots, X_n) - H(Z_1, Z_2, \dots, Z_n) \end{aligned} \quad (16)$$

$$\geq H(X_1, X_2, \dots, X_n) - \sum H(Z_i) \quad (17)$$

$$= H(X_1, X_2, \dots, X_n) - nH(p) \quad (18)$$

$$= n - nH(p), \quad (19)$$

if  $X_1, X_2, \dots, X_n$  are chosen i.i.d.  $\sim \text{Bern}(\frac{1}{2})$ . The capacity of the channel with memory over  $n$  uses of the channel is

$$nC^{(n)} = \max_{p(x_1, x_2, \dots, x_n)} I(X_1, X_2, \dots, X_n; Y_1, Y_2, \dots, Y_n) \quad (20)$$

$$\geq I(X_1, X_2, \dots, X_n; Y_1, Y_2, \dots, Y_n)_{p(x_1, x_2, \dots, x_n) = \text{Bern}(\frac{1}{2})} \quad (21)$$

$$\geq n(1 - H(p)) \quad (22)$$

$$= nC. \quad (23)$$

Hence channels with memory have higher capacity. The intuitive explanation for this result is that the correlation between the noise decreases the effective noise; one could use the information from the past samples of the noise to combat the present noise.

**7.5 Using two channels at once.** Suppose we are given two channels,  $(\mathcal{X}_1, p(y_1|x_1), \mathcal{Y}_1)$  and  $(\mathcal{X}_2, p(y_2|x_2), \mathcal{Y}_2)$ , which we can use at the same time. We can define the product channel as the channel,  $(\mathcal{X}_1 \times \mathcal{X}_2, p(y_1, y_2|x_1, x_2) = p(y_1|x_1)p(y_2|x_2), \mathcal{Y}_1 \times \mathcal{Y}_2)$ . To find the capacity of the product channel, we must find the distribution  $p(x_1, x_2)$  on the input alphabet  $\mathcal{X}_1 \times \mathcal{X}_2$  that maximizes  $I(X_1, X_2; Y_1, Y_2)$ . Since the joint distribution

$$p(x_1, x_2, y_1, y_2) = p(x_1, x_2)p(y_1|x_1)p(y_2|x_2), \quad (24)$$

$Y_1 \rightarrow X_1 \rightarrow X_2 \rightarrow Y_2$  forms a Markov chain and therefore

$$I(X_1, X_2; Y_1, Y_2) = H(Y_1, Y_2) - H(Y_1, Y_2 | X_1, X_2) \quad (25)$$

$$= H(Y_1, Y_2) - H(Y_1 | X_1, X_2) - H(Y_2 | X_1, X_2) \quad (26)$$

$$= H(Y_1, Y_2) - H(Y_1 | X_1) - H(Y_2 | X_2) \quad (27)$$

$$\leq H(Y_1) + H(Y_2) - H(Y_1 | X_1) - H(Y_2 | X_2) \quad (28)$$

$$= I(X_1; Y_1) + I(X_2; Y_2), \quad (29)$$

where (26) and (27) follow from Markovity, and we have equality in (28) if  $Y_1$  and  $Y_2$  are independent. Equality occurs when  $X_1$  and  $X_2$  are independent. Hence

$$C = \max_{p(x_1, x_2)} I(X_1, X_2; Y_1, Y_2) \quad (30)$$

$$\leq \max_{p(x_1, x_2)} I(X_1; Y_1) + \max_{p(x_1, x_2)} I(X_2; Y_2) \quad (31)$$

$$= \max_{p(x_1)} I(X_1; Y_1) + \max_{p(x_2)} I(X_2; Y_2) \quad (32)$$

$$= C_1 + C_2. \quad (33)$$

with equality iff  $p(x_1, x_2) = p^*(x_1)p^*(x_2)$  and  $p^*(x_1)$  and  $p^*(x_2)$  are the distributions that maximize  $C_1$  and  $C_2$  respectively.

7.8 *The Z channel.* First we express  $I(X; Y)$ , the mutual information between the input and output of the Z-channel, as a function of  $x = \Pr(X = 1)$ :

$$\begin{aligned} H(Y|X) &= \Pr(X = 0) \cdot 0 + \Pr(X = 1) \cdot 1 = x \\ H(Y) &= \mathcal{H}(\Pr(Y = 1)) = \mathcal{H}(x/2) \\ I(X; Y) &= H(Y) - H(Y|X) = \mathcal{H}(x/2) - x \end{aligned}$$

Since  $I(X; Y) = 0$  when  $x = 0$  and  $x = 1$ , the maximum mutual information is obtained for some value of  $x$  such that  $0 < x < 1$ .

Using elementary calculus, we determine that

$$\frac{d}{dx} I(X; Y) = \frac{1}{2} \log_2 \frac{1 - x/2}{x/2} - 1,$$

which is equal to zero for  $x = 2/5$ . (It is reasonable that  $\Pr(X = 1) < 1/2$  because  $X = 1$  is the noisy input to the channel.) So the capacity of the Z-channel in bits is  $H(1/5) - 2/5 = 0.722 - 0.4 = 0.322$ .

7.28 *Choice of channels.*

We can communicate information to the destination via the \*choice\* of channel (in addition to the information we can transmit over either channel). Furthermore, since the output alphabets are distinct, the information embedded in the choice of channel is received perfectly.

Let  $Z$  be a random variable with alphabet  $\{1, 2\}$ , where  $Z = 1$  indicates that channel 1 is selected for the current transmission,  $Z = 2$  indicates channel 2 is selected, and  $\Pr\{Z = 1\} = \alpha$ . Let  $X$  be a random variable (representing the channel input) with alphabet  $\{\mathcal{X}_1 \cup \mathcal{X}_2\}$ , and similarly let  $Y$  be a random variable (representing the channel output) with alphabet  $\{\mathcal{Y}_1 \cup \mathcal{Y}_2\}$ .

(a) We communicate to the destination via  $Z$  and  $X$ . We have that

$$\begin{aligned} I(X, Z; Y) &= H(Z, X) - H(X, Z|Y) \\ &= H(Z) + H(X|Z) - H(X|Y) \\ &= H_2(\alpha) + \alpha H(X_1) + (1 - \alpha)H(X_2) - H(X|Y) \\ &= H_2(\alpha) + \alpha H(X_1) + (1 - \alpha)H(X_2) - \alpha H(X_1|Y_1) - (1 - \alpha)H(X_2|Y_2) \\ &= H_2(\alpha) + \alpha I(X_1; Y_1) + (1 - \alpha)I(X_2; Y_2). \end{aligned}$$

Now, we obtain the capacity by maximizing over the choice of input distribution, which involves maximizing over  $p(x_1)$ ,  $p(x_2)$  and  $\alpha$ . Note that  $I(X_1; Y_1)$  depends only on  $p(x_1)$  and similarly for  $I(X_2; Y_2)$  and  $p(x_2)$ . Thus,

$$C = \max_{\alpha} H_2(\alpha) + \alpha C_1 + (1 - \alpha)C_2.$$

Now, it is an exercise in calculus to show that this quantity is maximized when

$$\alpha = \frac{2^{C_1 - C_2}}{1 + 2^{C_1 - C_2}},$$

and that

$$2^C = 2^{C_1} + 2^{C_2}$$

for the optimal choice of  $\alpha$ .

- (b) We have that the effective alphabet size for noise-free communication is the sum of the effective alphabets of the two constituent channels. In particular, note that the condition that the output symbols be disjoint is analogous to the condition in Problem 2.10 that the random variables have disjoint support sets.
- (c) The capacity of the BSC is  $1 - H_2(p)$ , and that of the single-input single-output channel is zero, thus

$$C = \log_2(1 + 2^{1 - H_2(p)}).$$

## References

- [1] R. Gallager, "Variations on a theme by Huffman," *IEEE Trans. Inform. Theory*, vol. IT-24, pp. 668–674, 1978.