Model Answers to Exercise 3 of October 1, 2014

http://www.isi.ee.ethz.ch/teaching/courses/it1/

Problem 1  
Entropy is Submodular

Fix $S, T \in 2^\Omega$. The proof is accomplished by the following chain of (in)equalities:

\[
H(S) + H(T) \overset{(a)}{=} H(S \cap T, S) + H(T) \\
\overset{(b)}{=} H(S \cap T) + H(S | S \cap T) + H(T) \\
\overset{(c)}{\geq} H(S \cap T) + H(S | T) + H(T) \\
\overset{(d)}{=} H(S \cap T) + H(S, T) \\
\overset{(e)}{=} H(S \cap T) + H(S \cup T),
\]

where (a) is true since $S \cap T \subseteq S$ and the chance variables in the collection $S \cap T$ are thus deterministic given the ones in $S$, (b) follows from the chain rule for entropy, (c) is true because $S \cap T \subseteq T$ and because conditioning cannot increase entropy, (d) follows from the chain rule for entropy, and (e) holds because there is a one-to-one relationship between the chance variables in the collection $S \cup T$ and the ones in $S$ and $T$.

Problem 2  
Pure Randomness and Bent Coins

a) i) We make use of the chain rule:

\[
H(X_1, X_2, \ldots, X_n) = \sum_{i=1}^{n} H(X_i | X_{i-1}, \ldots, X_1).
\]

Since in our case the random variables $X_1, \ldots, X_n$ are IID, we have

\[
H(X_1, X_2, \ldots, X_n) = \sum_{i=1}^{n} H(X_i) = n H_b(p),
\]

where the last equation follows from the fact that the $X_i$’s are binary random variables with probability $p$.

ii) The random vector $W \triangleq (Z_1, Z_2, \ldots, Z_K, K)$ is a function of the random vector $Y \triangleq (X_1, X_2, \ldots, X_n)$, i.e., $W = f(Y)$. However, we know that applying a function to a random vector never increases the uncertainty. To prove this, consider the following sequence of (in)equalities:

\[
H(Y, f(Y)) = H(Y) + H(f(Y) | Y) = H(Y); \\
H(Y, f(Y)) = H(f(Y)) + H(Y | f(Y)) \geq H(f(Y)).
\]

Thus, $H(X_1, X_2, \ldots, X_n) \geq H(Z_1, Z_2, \ldots, Z_K, K)$. 

© Amos Lapidoth, 2014/2015
iii) This follows from the chain rule.

iv) For a given $K$, $Z_1, \ldots, Z_K$ are pure random bits, i.e., $H(Z_i|K = k) = 1$ bit and all $Z_i$’s are independent of each other. Therefore,

$$H(Z_1, Z_2, \ldots, Z_K|K) = \sum_k P_K(k) \cdot H(Z_1, Z_2, \ldots, Z_K|K = k)$$

$$= \sum_k P_K(k) \sum_{i=1}^k H(Z_i|K = k)$$

$$= \sum_k P_K(k) \sum_{i=1}^k 1 \text{ bit}$$

$$= \sum_k P_K(k) k$$

$$= E[K] \text{ bits.}$$

v) This follows from the nonnegativity of entropy.

b) Since we do not know $p$, the only way to generate pure random bits is to use the fact that all sequences with the same number of ones are equally likely. For example, the sequences 0001, 0010, 0100 and 1000 are equally likely and can be used to generate two pure random bits. The sequences 0011, 0101, 1001, 0110, 1010, and 1100 are also equally likely, however one cannot produce $\log_2 6$ pure random bits as six is not a power of 2. Here, we can choose four out of the six sequences and produce two pure random bits, and then take the remaining two sequences for producing one random bit. An example of a mapping to generate random bits is

$$0000 \rightarrow \Lambda$$
$$0001 \rightarrow 00 \quad 0010 \rightarrow 01 \quad 0100 \rightarrow 10 \quad 1000 \rightarrow 11$$
$$0011 \rightarrow 00 \quad 0110 \rightarrow 01 \quad 1100 \rightarrow 10 \quad 1010 \rightarrow 11$$
$$1010 \rightarrow 0 \quad 0101 \rightarrow 1$$
$$1110 \rightarrow 11 \quad 1101 \rightarrow 10 \quad 1011 \rightarrow 01 \quad 0111 \rightarrow 00$$
$$1111 \rightarrow \Lambda$$

The resulting expected number of produced random bits is

$$E[K] = (1 - p)^4 \cdot 0 + 4p(1 - p)^3 \cdot 2 + 4p^2(1 - p)^2 \cdot 2 + 2p^2(1 - p)^2 \cdot 1$$

$$+ 4p^3(1 - p) \cdot 2 + p^4 \cdot 0$$

$$= 8p(1 - p)^3 + 10p^2(1 - p)^2 + 8p^3(1 - p).$$

For example, for $p \approx \frac{1}{2}$, the expected number of pure random bits is close to 1.625. This is substantially less than the 4 pure random bits that could be generated if $p$ were exactly $\frac{1}{2}$. Even though this was not asked for, and for the interested reader only, we will now try to analyze the efficiency of this scheme of generating random bits for long sequences of bent coin flips. Let $n$ be the number of bent coin flips. The algorithm that we will use is the obvious extension of the above method of generating pure bits using the fact that all sequences with the same number of ones are equally likely.

Consider all sequences with $j$ ones. There are $\binom{n}{j}$ such sequences, which are all equally likely. If $\binom{n}{j}$ were a power of 2, then we could generate $\log_2 \binom{n}{j}$ pure random bits from such a set. However, in the general case, $\binom{n}{j}$ is not a power of 2 and the best we can do is to divide the set of $\binom{n}{j}$ elements into subsets of sizes which are powers of 2. The largest set would have a size $2^{\lceil \log_2 \binom{n}{j} \rceil}$ and could be used to generate $\lfloor \log_2 \binom{n}{j} \rfloor$ random bits. We could divide the
remaining elements into the largest set which is a power of 2, etc. Intuitively it is clear that
the worst case would occur when \( \binom{n}{j} = 2^{l+1} - 1 \) (where \( l \) is an integer), in which case the
subsets would be of sizes \( 2^l, 2^{l-1}, 2^{l-2}, \ldots, 1 \).

Instead of analyzing the scheme exactly, we will just find a lower bound on the number of
random bits generated from a set of size \( \binom{n}{j} \). Let \( l \triangleq \lfloor \log_2 \binom{n}{j} \rfloor \). Then in the worst case half
of the elements belong to a set of size \( 2^l \) and would generate \( l \) random bits, one fourth of the
elements belong to a set of size \( 2^{l-1} \) and generate \( l - 1 \) random bits, etc. On the average, the
number of bits generated is

\[
\mathbb{E}[K \mid j \text{ 1's in a sequence}] \geq \frac{1}{2} \cdot l + \frac{1}{4}(l-1) + \frac{1}{8}(l-2) + \cdots + \frac{1}{2^l} \cdot 1
\]

\[
= \left( \frac{1}{2} + \frac{1}{4} + \cdots + \frac{1}{2^l} \right) + \left( \frac{1}{4} + \frac{1}{4} + \cdots + \frac{1}{4} \right) + \cdots + \left( \frac{1}{2^{l-1}} + \frac{1}{2^{l-1}} \right) + \left( \frac{1}{2^l} \right)
\]

\[
= \left( 1 - \frac{1}{2^l} \right) + \left( 1 - \frac{1}{2^{l-1}} \right) + \cdots + \left( 1 - \frac{1}{4} \right) + \left( 1 - \frac{1}{2} \right)
\]

\[
= l - \sum_{i=1}^{l} \left( \frac{1}{2} \right)^i
\]

\[
= l - \frac{1}{2^{l}} - \frac{1}{2^{l-1}}
\]

\[
= l \cdot 1 + 2^{-l}
\]

\[
\geq l - 1.
\]

Hence the fact that \( \binom{n}{j} \) is not a power of 2 will cost at most 1 bit on the average in the
number of random bits that are produced.

Therefore, similarly to our first example with \( n = 4 \), the expected number of pure random
bits produced by this algorithm is

\[
\mathbb{E}[K] \geq \sum_{j=0}^{n} \binom{n}{j} p^j (1 - p)^{n-j} \left[ \log_2 \left( \binom{n}{j} \right) - 1 \right]
\]

\[
\geq \sum_{j=0}^{n} \binom{n}{j} p^j (1 - p)^{n-j} \left( \log_2 \left( \binom{n}{j} \right) - 2 \right)
\]

\[
= \sum_{j=0}^{n} \binom{n}{j} p^j (1 - p)^{n-j} \log_2 \left( \binom{n}{j} \right) - 2 \sum_{j=0}^{n} \binom{n}{j} p^j (1 - p)^{n-j}
\]

\[
= \sum_{j=0}^{n} \binom{n}{j} p^j (1 - p)^{n-j} \log_2 \binom{n}{j} - 2
\]

\[
\geq \sum_{n(p-\epsilon) \leq j \leq n(p+\epsilon)} \binom{n}{j} p^j (1 - p)^{n-j} \log_2 \binom{n}{j} - 2,
\]

where the last inequality holds because we reduce the number of addends in the summation.
This is done because the weak law of large numbers says that for a sufficiently large \( n \) the probability,
that the number of 1’s in the sequence is close to \( n \cdot p \), is almost 1, i.e., the
addends for small \( j \)'s and the addends for large \( j \)'s (i.e., \( j \approx n \)) do not contribute much to the sum. In this case \( j \) is also large if \( n \) is large, and we can use Stirling’s approximation:

\[
\sqrt{2\pi n} \cdot n^n \cdot e^{-n} < n! < \sqrt{2\pi n} \cdot n^n \cdot e^{-n - \frac{1}{2n}}
\]

Thus,

\[
\binom{n}{j} = \frac{n!}{j!(n-j)!} \geq \sqrt{\frac{2\pi j \cdot j^j \cdot e^{-j + \frac{1}{12j}}}{\sqrt{2\pi n} \cdot n^n \cdot e^{-n}}} \cdot \left( (n-j)(n-j) \right)^{-(n-j) + \frac{1}{12(n-j)}}
\]

\[
= \sqrt{\frac{n}{2\pi j(n-j)}} \cdot e^{-n - j + \frac{1}{12j} + n - j - \frac{1}{12(n-j)}} \cdot \frac{2n \log_2 n}{2j \log_2 j \cdot 2(n-j) \log_2(n-j)}
\]

(2)

where the exponent can be rewritten as

\[
= n \log_2 n - j \log_2 j - (n-j) \log_2(n-j)
\]

\[
= n \left( \log_2 n - \frac{j}{n} \log_2 j - \frac{n-j}{n} \log_2(n-j) \right)
\]

\[
= n \left( \log_2 n - \frac{j}{n} \log_2 j + \frac{j}{n} \log_2 n - \frac{n-j}{n} \log_2(n-j) \right) + \frac{n-j}{n} \log_2 n - \frac{j}{n} \log_2 n
\]

\[
= n \left( -\frac{j}{n} \log_2 j - \frac{n-j}{n} \log_2(n-j) + \frac{n-j}{n} \log_2 n - \frac{j}{n} \log_2 n \right)
\]

\[
= n H_b \left( \frac{j}{n} \right).
\]

Using this in inequality (2) and choosing a sufficiently large \( \delta_1 \geq 0 \), we get

\[
\binom{n}{j} \geq \sqrt{\frac{n}{2\pi j(n-j)}} \cdot e^{\frac{n-j}{12j(n-j)}} \cdot 2^{n H_b \left( \frac{j}{n} \right)} \geq 2^n \left( H_b \left( \frac{j}{n} \right) - \delta_1 \right).
\]

Since we restrict the summation over \( j \) to \( j \) that are close to \( n \cdot p \), we have that \( \frac{j}{n} \) is close to \( p \), and hence we can find a \( \delta_2 \geq 0 \) such that

\[
\binom{n}{j} \geq 2^n \left( H_b \left( \frac{j}{n} \right) - \delta_1 \right) \geq 2^n \left( H_b(p) - \delta_1 - \delta_2 \right) \geq 2^n \left( H_b(p) - 2\delta \right)
\]

where \( \delta = \max \{ \delta_1, \delta_2 \} \). We use this approximation in inequality (1):

\[
E[K] \geq \sum_{n(p-\epsilon) \leq j \leq n(p+\epsilon)} \binom{n}{j} p^j (1-p)^{n-j} 2^{n H_b(p) - 2\delta} - 2
\]

\[
= n \left( H_b(p) - 2\delta \right) \sum_{n(p-\epsilon) \leq j \leq n(p+\epsilon)} \binom{n}{j} p^j (1-p)^{n-j} - 2
\]

\[
\geq n \left( H_b(p) - 2\delta \right) (1-\epsilon) - 2,
\]

where in the last inequality we assume that \( n \) is large enough so that the probability that \( n(p-\epsilon) \leq j \leq n(p+\epsilon) \) is greater than \( 1 - \epsilon \) for an \( \epsilon \geq 0 \). This result is very good since \( n H_b(p) \) is an upper bound on the number of pure random bits that can be produced from the bent coin sequence. Hence, we see that for large \( n \), our scheme will work efficiently.
Problem 3

*Conditional vs. Unconditional Mutual Information*

a) We want to find a situation where the information “about X”, which you “get from Y”, is greater than the information gained from Y about X when we already know Z. A simple example of such a situation is if X is the result of a fair coin flip, and Y = X and Z = Y. In this case,

\[ I(X; Y) = H(X) - H(X|Y) = H(X) = 1 \text{ bit} \]

and

\[ I(X; Y|Z) = H(X|Z) - H(X|Y, Z) = 0 \text{ bits}, \]

so that \( I(X; Y|Z) < I(X; Y) \).

More generally we can say that if X and Y are dependent and either \( Z = X \) or \( Z = Y \), then

\[ I(X; Y) = H(X) - H(X|Y) > 0. \]

Also,

\[ I(X; Y|Z) = H(X|Z) - H(X|Y, Z) = 0 < I(X; Y). \]

The last equation follows from the fact that either the two addends are both zero (if \( Z = X \)), or that \( H(X|Z) = H(X|Y, Z) \) (if \( Z = Y \)).

b) In this example Z is supposed to increase the information you get from Y about X. Let X and Y be two independent binary random variables with \( P_X(x=0) = P_X(x=1) = P_Y(y=0) = P_Y(y=1) = \frac{1}{2} \). And let \( Z = X + Y \). Then

\[ I(X; Y) = H(X) - H(X|Y) = H(X) - H(X) = 0 \text{ bits} \]

and

\[ I(X; Y|Z) = H(X|Z) - H(X|Y, Z) = H(X|Z) = \frac{1}{2} \text{ bits}, \]

so that \( I(X; Y|Z) > I(X; Y) \). More generally, the inequality is still valid if X and Y are any independent random variables and \( X = g(Y, Z) \) is a function that allows to uniquely determine X given Z and Y but not so if only Z is provided. In this case, \( H(X|Z) > 0 \) and \( H(X|Y, Z) = 0 \).

Problem 4

*Classes of Codes*

a) No, the code is not instantaneous, since the first codeword, 0, is a prefix of the second codeword, 01.

b) Yes, the code is uniquely decodable. Given a sequence of codewords, first isolate occurrences of 01 (i.e., find all the 1’s) and then parse the rest into 0’s.

c) Yes, all uniquely decodable codes are nonsingular.
Problem 5

**Slackness in Kraft’s Inequality**

a) The codewords of an instantaneous code can be assigned to internal nodes or leaves of a binary tree of depth \( l_{\text{max}} = \max\{l_1, l_2, \ldots, l_m\} \) such that a codeword of length \( l_i \) is assigned to an internal node at depth \( l_i \) and all the children of this node cannot represent any other codeword. We will describe this later fact by saying that the \( 2^{l_{\text{max}}-l_i} \) descending leaves are shadowed, which is depicted in Fig. 1 i) using dotted lines and gray colored leaves. (In case the node representing the codeword is a leaf itself, we will also say that the leaf is shadowed.)

![Diagram of shadowed leaves](image)

Figure 1: A tree representing a prefix-free code satisfying Kraft’s inequality with strict inequality is depicted in i), and a tree corresponding to a better prefix-free code satisfying Kraft’s inequality with equality is shown in ii). Nodes corresponding to codewords are black, shadowed leaves (unless they correspond to codewords) are gray and unshadowed leaves are white.

The tree consists of \( 2^{l_{\text{max}}} \) leaves, and the total number of leaves which are shadowed is

\[
\sum_{i=1}^{m} 2^{l_{\text{max}}-l_i} = 2^{l_{\text{max}}} \sum_{i=1}^{m} 2^{-l_i} \overset{(*)}{<} 2^{l_{\text{max}}} \cdot 1 = 2^{l_{\text{max}}},
\]

where the strict inequality (\( * \)) holds by assumption. Hence there are leaves which are not shadowed by any codeword, i.e., on the path from the leaf to the root there is no codeword assigned. Consequently, we can improve our code in the following way:

- choose one of the leaves that is not shadowed and follow its path up to the root as long as the current node has only descendant leaves which are not shadowed;
- as soon as you encounter an internal node for which a part of the descendants is shadowed, remove that internal node and replace it by the child which belongs to the path leading to the shadowed leaf.

(For an illustration see the transition from Fig. 1 i) to ii).) This clearly reduces the length of the codewords.

b) The conclusion generally does not hold for \( D > 2 \). A simple example is to choose \( D = 3 \) and \( m = 2 \). Clearly, the “best” ternary code for a source with 2 possible outcomes uses one symbol to describe each outcome. Then we have \( l_1 = l_2 = 1 \) and

\[
\sum_{i=1}^{2} 3^{-l_i} = \frac{1}{3} + \frac{1}{3} = \frac{2}{3} < 1.
\]

Thus for our code Kraft’s inequality holds with strict inequality. Nevertheless, it is not possible to find a deterministically better code for the described source.

© Amos Lapidoth, 2014/2015