Solutions to Practice Midterm (Spring 2016)

1. Binary codes (30 points). Consider a source that emits five symbols \{A, B, C, D, E\} with probabilities 0.3, 0.3, 0.2, 0.1, and 0.1, respectively.

(a) Construct a binary Huffman code for this source, taking one source symbol at a time. What is the average codeword length for this code?

(b) Repeat part (a) for a binary Shannon–Fano code taking one source symbol at a time.

(c) Construct a probability distribution \( p = (p_A, p_B, p_C, p_D, p_E) \) on \{A, B, C, D, E\}, for which the code that you constructed in part (a) has an average length equal to its binary entropy \( H(p) \).

Solution:

(a) One possible way to do the Huffman coding is shown below.

```
A  0.3
  0
   0.6

B  0.3
  1

C  0.2
  0
   0.4

D  0.1
  0
     0.2
     1

E  0.1
```
This gives the encoding

\[
\begin{align*}
A & \rightarrow 00 \\
B & \rightarrow 01 \\
C & \rightarrow 10 \\
D & \rightarrow 110 \\
E & \rightarrow 111
\end{align*}
\]

The average codeword length is given by

\[
\bar{L} = 2 \times (0.3 + 0.3 + 0.2) + 3 \times (0.1 + 0.1)
= 2.2 \text{ bits per symbol.}
\]

Alternatively, the average length can be computed by adding the probabilities at the internal nodes, so we have

\[
\bar{L} = 1.0 + 0.6 + 0.4 + 0.2
= 2.2 \text{ bits per symbol.}
\]
(b) The procedure for constructing a binary Shannon–Fano code is shown below.

![Shannon–Fano Code Diagram]

This gives the encoding

\[
A \rightarrow 00 \\
B \rightarrow 01 \\
C \rightarrow 10 \\
D \rightarrow 110 \\
E \rightarrow 111,
\]

which is the same as the Huffman code. Thus, the average length is the same. Similar to part (a), the average length can also be computed by adding the probabilities at the internal nodes, giving

\[
\bar{L} = 1.0 + 0.6 + 0.4 + 0.2 \\
= 2.2 \text{ bits per symbol.}
\]
(c) We see that for the Huffman code we constructed, the codeword lengths satisfy
\[ \sum_i 2^{-l_i} = 1. \]
Hence, if we have a probability distribution with \( p_i = 2^{-l_i} \), our code
will have average codeword length given by
\[
\bar{L} = \sum_i p_i l_i \\
= -\sum_i p_i \log p_i \quad \text{(since } p_i = 2^{-l_i} \text{)} \\
= H(p),
\]
i.e., the entropy of the distribution.
This shows that the required probabilities are \( p_A = p_B = p_C = 1/4, p_D = p_E = 1/8 \).
This gives an average codeword length of \( \bar{L} = 2.25 \) bits per symbol, which is the
same as \( H(p_A, p_B, p_C, p_D, p_E) \).

2. **One-bit quantizer (30 points).** Let \( X \) be drawn according to the pdf

\[ f_X(x) = \begin{cases} 
2x, & 0 \leq x \leq 1, \\
0, & \text{otherwise.}
\end{cases} \]

(a) Given the quantization regions \( R_1 = \{ x : 0 \leq x < b \} \) and \( R_2 = \{ x : b \leq x \leq 1 \} \),
find the quantization points \( a_1 \in R_1 \) and \( a_2 \in R_2 \) that minimize the mean squared
error (MSE) in terms of \( b \).

(b) Given the quantization points \( a_1 < a_2 \), find the quantization regions that minimize
the MSE in terms of \( a_1 \) and \( a_2 \).

(c) Using parts (a) and (b), find the optimal quantizer by specifying \( a_1, a_2, \) and \( b \) that
minimize the MSE.

**Solution:**

4
(a) Using the centroid condition, we have

\[
a_1 = \frac{\int_0^b 2x^2 \, dx}{\int_0^b 2x \, dx} = \frac{(2/3)b^3}{b^2} = \frac{2}{3}b
\]

and

\[
a_2 = \frac{\int_b^1 2x^2 \, dx}{\int_b^1 2x \, dx} = \frac{(2/3)(1-b^3)}{1-b^2} = \frac{2(b^2 + b + 1)}{3(b+1)}.
\]

(b) Using the Voronoi condition, the quantization regions are given by \(R_1 = [0, b)\) and \(R_2 = [b, 1]\), where

\[
b = \frac{a_1 + a_2}{2}.
\]

(c) Plugging the expressions for \(a_1\) and \(a_2\) computed in part (a) into the expression in part (b), we have

\[
2b = \frac{2}{3}b + \frac{2(b^2 + b + 1)}{3(b+1)}
\]

\[
\Rightarrow \frac{4}{3}b = \frac{2(b^2 + b + 1)}{3(b+1)}
\]

\[
\Rightarrow 2b = \frac{b^2 + b + 1}{b+1}
\]

\[
\Rightarrow 2b^2 + 2b = b^2 + b + 1
\]

\[
\Rightarrow b^2 + b - 1 = 0
\]

\[
\Rightarrow b = \frac{\sqrt{5} - 1}{2}.
\]

Plugging this value into the expressions in part (a), we have

\[
a_1 = \frac{\sqrt{5} - 1}{3} \text{ and } a_2 = \frac{2(\sqrt{5} - 1)}{3}.
\]

3. Source coding for emergency (20 points). Consider a source that emits six symbols \(\{A, B, C, D, E, F\}\) with probabilities 0.3, 0.2, 0.2, 0.1, 0.1, and 0.1, respectively. Here
the symbol $F$ only has the probability 0.1, but it is meant for an emergency message, say, “Fire!”, and hence we would like to spend few bits in encoding $F$.

(a) Design a binary instantaneous code that takes one source symbol at a time with the shortest possible average codeword length under the constraint that $F$ should be mapped to the codeword “0” of length 1.

(b) (Difficult.) Now assume that each source symbol $i$ has some associated cost per letter $c_i$ of the codeword. The normal symbols $A, B, C, D, E$ has a cost of $c_i = 1$ per codeword letter, while the emergency symbol $F$ has a cost of $c_i = 4$ per code letter. Hence, if the codeword for symbol $i$ has the length $i$, then the cost spent for the symbol is $c_il_i$. For example, the code that maps

\[
\begin{align*}
    A & \rightarrow 0001 \\
    B & \rightarrow 00000 \\
    C & \rightarrow 1100 \\
    D & \rightarrow 111 \\
    E & \rightarrow 01 \\
    F & \rightarrow 10
\end{align*}
\]

spends the cost of 4, 5, 4, 3, 2, and 8, respectively, for the symbols $A, B, C, D, E$, and $F$. Design a binary instantaneous code that minimizes the average cost

$$\sum_i p_i c_i l_i$$

of sending a symbol.

Solution:

(a) Since $F$ is mapped to “0” and the code has to be instantaneous, all other codewords must start with “1”. In order to ensure that we have the shortest possible average codeword length, we can perform Huffman coding on the source symbols other than $F$, and finally add “1” at the beginning of these other codewords.

Note that the total probability of the symbols other than $F$ is not 1; however, the Huffman coding algorithm does not require the numbers associated with the symbols to sum up to 1; any set of non-negative weights can be used, and the resulting encoding will then give the code with the shortest “average” codeword length, where the average is computed using the corresponding weights.
One way to perform the Huffman coding is as follows.

This, together with the preceding discussion, gives the encoding

\[
A \rightarrow 100 \\
B \rightarrow 101 \\
C \rightarrow 110 \\
D \rightarrow 1110 \\
E \rightarrow 1111 \\
F \rightarrow 0
\]

The average codeword length is given by

\[
\bar{L} = 3 \times (0.3 + 0.2 + 0.2) + 4 \times (0.1 + 0.1) + 1 \times 0.1 \\
= 3.0 \text{ bits per symbol.}
\]

Alternatively, the average codeword length can also be computed as

\[
\bar{L} = 1 + (\text{sum of probabilities at internal nodes of the Huffman coding tree}) \\
= 1 + 0.9 + 0.5 + 0.4 + 0.2 \\
= 3.0 \text{ bits per symbol.}
\]

(b) We are required to minimize \(\sum p_i c_i l_i = \sum q_i l_i\), where \(q_i = p_i c_i\). As a consequence of the argument provided in part (a), in order to minimize the average cost, we can use Huffman coding with the probability of a symbol replaced by the
corresponding weight \( q_i \). The procedure is shown below.

This gives the encoding

\[
A \rightarrow 00 \\
B \rightarrow 01 \\
C \rightarrow 110 \\
D \rightarrow 1110 \\
E \rightarrow 1111 \\
F \rightarrow 10
\]

The average cost is given by

\[
\bar{C} = 2 \times (0.3 + 0.2 + 0.4) + 3 \times 0.2 + 4 \times (0.1 + 0.1) = 3.2 \text{ per symbol.}
\]

Alternatively, the average cost can be computed as the sum of the weights at the internal nodes, and is thus given by

\[
\bar{C} = 1.3 + 0.8 + 0.5 + 0.4 + 0.2 = 3.2 \text{ per symbol.}
\]
1. Consider a source which produces an i.i.d. sequence of symbols from the alphabet \{A, B, C, D, E, F, G\} with probabilities \{0.22, 0.35, 0.15, 0.09, 0.09, 0.05, 0.05\} respectively.

a. Find binary Huffman codes and compute the average number of binary code symbols per source symbol. **(18 points)**

b. Repeat part a. except find Shannon-Fano codes instead of Huffman codes. **(15 points)**

**Solution:**

a. Applying the Huffman coding algorithm, we obtain the following code.

<table>
<thead>
<tr>
<th>Code</th>
<th>Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>B</td>
</tr>
<tr>
<td>10</td>
<td>A</td>
</tr>
<tr>
<td>010</td>
<td>C</td>
</tr>
<tr>
<td>110</td>
<td>D</td>
</tr>
<tr>
<td>111</td>
<td>E</td>
</tr>
<tr>
<td>0110</td>
<td>F</td>
</tr>
<tr>
<td>0111</td>
<td>G</td>
</tr>
</tbody>
</table>

The average number of binary code symbols per source symbol is

$$\bar{L}_H = 2 \times (0.35 + 0.22) + 3 \times (0.15 + 0.09 + 0.09) + 4 \times (0.05 + 0.05) = 2.53.$$ 

b. Applying the Shannon–Fano coding algorithm, we have the following code.

<table>
<thead>
<tr>
<th>Code</th>
<th>Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.35</td>
<td>0</td>
</tr>
<tr>
<td>0.22</td>
<td>1</td>
</tr>
<tr>
<td>0.15</td>
<td>0</td>
</tr>
<tr>
<td>0.09</td>
<td>0</td>
</tr>
<tr>
<td>0.09</td>
<td>1</td>
</tr>
<tr>
<td>0.05</td>
<td>1</td>
</tr>
<tr>
<td>0.05</td>
<td>1</td>
</tr>
</tbody>
</table>

The average number of binary code symbols per source symbol is

$$\bar{L}_{S-F} = \bar{L}_H = 2.53.$$
2. Consider a binary source $A, B$ with $P(B) = \frac{1}{16}$.

a. Find a Tunstall code for the source that encodes the source phrases into binary code words of length 3. Compute the average number of binary code symbols per source symbol. (22 points)

**Hint:** For all $n \leq 40, (\frac{15}{16})^n \geq \frac{1}{16}$. Also, $(\frac{15}{16})^7 = 0.63$.

**Hint:** For all $a \in \mathbb{R}, N \in \mathbb{N}, \sum_{i=0}^{N} a^i = \frac{1-a^{N+1}}{1-a}$.

b. (Bonus) Can you generalize the above construction to Tunstall codes of length 4 and 5? (10 BONUS points)

Solution:

a. We apply the Tunstall coding algorithm and get the following code, where we denote $A^n = A \cdot \cdots \cdot A$.

\[
\begin{array}{c}
\left(\frac{15}{16}\right)^5 A^5 \xrightarrow{A^4} A^4 B \xrightarrow{154} \\
\left(\frac{15}{16}\right)^6 A^6 \xrightarrow{A^5} A^6 B \xrightarrow{156} \\
\left(\frac{15}{16}\right)^7 A^7 \\
\end{array}
\]

The average source symbol length in the dictionary is

\[
\mathbb{E}[L] = \frac{1}{16} + \frac{15}{16^2} \times 2 + \frac{15^2}{16^3} \times 3 + \frac{15^3}{16^4} \times 4 + \frac{15^4}{16^5} \times 5 + \frac{15^5}{16^6} \times 6 + \left(\frac{15^6}{16^7} + \frac{15^7}{16^7}\right) \times 7 = 5.816.
\]

The average number of code symbols per source symbol is $\frac{3}{\mathbb{E}[L]} = 0.516$.

b. Since $(\frac{15}{16})^{14} \geq (\frac{15}{16})^{30} \geq \frac{1}{16}$, we will always split the $A^n$ branch in the dictionary tree as in part a. Therefore, the Tunstall code of length 4 and 5 are given as follows.

\[
\begin{array}{c|c|c}
B & \text{Length-4 code} & A^8 B \\
AB & 0000 & 1000 \\
AAB & 0010 & A^9 B \\
A^3 B & 0011 & A^{10} B \\
A^4 B & 0100 & A^{11} B \\
A^5 B & 0101 & A^{12} B \\
A^6 B & 0110 & A^{13} B \\
A^7 B & 0111 & A^{14} B \\
\end{array}
\]
3. Consider the source given in Problem 2 for blocks of 8 symbols at a time.

   a. Compute the probability that a block only consists of “A”s. \((12 \text{ points})\)
      
      **Hint:** \(\left(1 - \frac{1}{n}\right)^2 \approx e^{-1/2} \approx 0.6\)
   
   b. Construct the most efficient simple prefix-free code consisting of 1 codeword of length 1 and 255 codewords of length \(m\). What is \(m\)? What is the average codeword length? Compute the average bits per symbol. \((22 \text{ points})\)
   
   c. Consider the set \(B\) of symbol blocks consisting of one and only one \(B\). What is the probability of set \(B\)? What is the minimum number of bits required to represent the blocks in \(B\)? \((15 \text{ points})\)
   
   d. Construct a simple prefix-free yet efficient coding scheme with three types of codewords: a) 1 codeword of length 1, b) codewords of length \(k\), where \(k\) is a fixed number larger than 1 and smaller than 8, and c) codewords of length \(l\), where \(l\) is a fixed number larger than or equal to 8. Compute the most efficient \(k\) and \(l\). \((26 \text{ points})\)
   
   e. Compute the average code length as well as average bits per source symbol for this code. \((10 \text{ points})\)

**Solution:**

   a. Applying the hint, we have \(P(A^8) = \left(\frac{15}{16}\right)^8 = \left(1 - \frac{1}{16}\right)^{\frac{16}{2}} \approx 0.6\).

   b. Since the codeword lengths of any prefix-free code satisfy Kraft’s inequality, we have

   \[2^{-1} + 255 \times 2^{-m} \leq 1.\]

   This implies that for the most efficient prefix-free code, \(m = 9\). Clearly we should assign length 1 to the most frequent symbol \(A^8\). The corresponding average codeword length is

   \[\bar{L} = 0.6 \times 1 + 0.4 \times 9 = 4.2.\]

   The average bits per symbol is

   \[\frac{\bar{L}}{8} = \frac{4.2}{8} = 0.525.\]
c. The set $\mathcal{B}$ contains 8 elements \{$BA^7, ABA^6, A^2BA^5, A^3BA^4, A^4BA^3, A^5BA^2, A^6BA, A^7B$\}. The probability of set $\mathcal{B}$ is
\[
P(\mathcal{B}) = 8 \times \frac{1}{16} \left(\frac{15}{16}\right)^7 = 0.318.
\]

d. We assign the codeword length as follows: $l(A^8) = 1$; for any $x \in \mathcal{B}, l(x) = k$; for the rest, $l(x) = l$. Then, by Kraft’s inequality, $k$ and $l$ should satisfy
\[
2^{-1} + 8 \times 2^{-k} + (2^8 - 1 - 8) \times 2^{-l} \leq 1.
\]

Considering the constraints that $1 < k \leq 8$ and $8 \leq l$, we choose $k = 5$ and $l = 10$.

e. The average code length of the code is
\[
\bar{L} = 0.6 \times 1 + 5 \times 0.318 + 10 \times (1 - 0.6 - 0.318) = 3.01.
\]

The average bits per source symbol for this code is
\[
\bar{L}/8 = 3.01/8 = 0.376.
\]