EE 231A<br>
Information Theory<br>
Instructor: Rick Wesel

Handout #6, Problem Set 3<br>
Tuesday, April14, 2020<br>
Due: Tuesday, April 21, 2020

106 pts

Reading: Chapter 5 and sections 13.3-13.4 of *Elements of Information Theory*

## Lectures 5: Single-Letter Source Codes, extension Codes, nonsingular, uniquely decodable, and instantaneous or prefix-free codes, Kraft and McMillan inequalities, minimum expected code length

1. (8 pts) *Code Analysis.*

   Consider a binary source code that employs the following codewords:

   0<br>
   01<br>
   001<br>
   001001<br>
   001001001

   (a) (4 pts) Is this code instantaneous? Does an instantaneous binary source code exist with these codeword lengths? (Justify your answers.)

   (b) (4 pts) Is this code uniquely decodeable? Does a uniquely decodeable binary source code exist with these codeword lengths? (Justify your answers.)

2. (8 pts) *Reversal.*

   Consider a binary source code that employs the following codewords:

   0<br>
   01<br>
   011<br>
   0111<br>
   01111<br>
   011111<br>
   111111

   (a) (4 pts) Is this code instantaneous? Does an instantaneous binary source code exist with these codeword lengths? (Justify your answers.)

   (b) (4 pts) Is this code uniquely decodeable? Does a uniquely decodeable binary source code exist with these codeword lengths? (Justify your answers.)

3. (8 pts) *Slackness in the Kraft inequality.*

   An instantaneous code has word lengths $l_1, l_2, \ldots, l_m$ which satisfy the strict inequality

$$\sum_{i=1}^{m} D^{-l_i} < 1.$$

   The code alphabet is $\mathcal{D} = \{0, 1, 2, \ldots, D-1\}$. Show that there exist sequences of code symbols in $\mathcal{D}^*$ which cannot be decoded into sequences of codewords. $\mathcal{D}^*$ is the set of finite length strings of symbols from the code alphabet.

4. (8 pts) *Optimal code lengths that require one bit above entropy.* The source coding theorem shows that the optimal instantaneous code for a random variable $X$ has an expected length less than $H(X)+1$. Give an example of a random variable for which the expected length of the optimal instantaneous code is close to $H(X)+1$, i.e., for any $\epsilon > 0$, construct a distribution for which the optimal code has $L > H(X) + 1 - \epsilon$.

## Lecture 6A: Huffman Coding.

5. (8 pts) *Huffman Code.*

   $X$ is distributed according to the following probability mass function:

$$\frac{1}{2} \quad \frac{1}{4} \quad \frac{1}{8} \quad \frac{1}{16} \quad \frac{1}{32} \quad \frac{1}{64} \quad \frac{1}{64}\,. \tag{1}$$

   Find a binary Huffman code for this distribution. Does it achieve the the corresponding entropy limit on compression?

6. (12 pts) *Non-binary Huffman Code?*

   $X$ is distributed according to the following probability mass function:

$$\frac{1}{3} \quad \frac{1}{3} \quad \frac{1}{9} \quad \frac{1}{9} \quad \frac{1}{27} \quad \frac{1}{27} \quad \frac{1}{27}\,. \tag{2}$$

   (a) (5 pts) Find a binary Huffman code for $X$. Compute its average length in bits.

   (b) (5 pts) Find a *ternary* Huffman code (for which there are three symbols instead of two, three brances in every Huffman step instead of two.) Compute it's average length (average number of *ternary symbols*, which are sometimes called trits.

   (c) (2 pts) Does the ternary Huffman code achieve the corresponding ternary entropy limit on compression? Which code is more efficient in this case, the ternary or the binary? Explain your answer.

7. (8 pts) *A sufficient set of Huffman codes .*

   (a) (4 pts) How many distinct binary Huffman codes does it take to handle all possible PMF's that have exactly three distinct outcomes with nonzero probability? In other words, what is the smallest number of Huffman codes so that for any three-outcome PMF one could label the branches to produce one of these codes?

   (b) (4 pts) How many distinct binary Huffman codes does it take to handle all possible PMF's that have exactly four distinct outcomes with nonzero probability?

8. (9 pts) *Codeword Lengths.*

   Consider a lossless compression code that uses five binary codewords with lengths $\{1, 2, 3, 4, 5\}$.

   (a) Prove that a uniquely decodable code exists with these codeword lengths. While you are at it, prove that a prefix free code exists with these codeword lengths.

   (b) Exhibit a prefix free code with these lengths.

   (c) Either give a PMF for which your code is a Huffman code, or explain why your code cannot be a Huffman code for any PMF.

## Lecture 6B: Optimality of Huffman Coding.

9. (9 pts) *Bad codes.* Which of these codes cannot be Huffman codes for any probability assignment?

   (a) $\{0, 10, 11\}$.

   (b) $\{00, 01, 10, 110\}$.

   (c) $\{01, 10\}$.

## Lectures 6C,D Shannon-Fano-Elias Coding and Arithmetic Coding

10. (18 pts) *Arithmetic Coding.*

In this problem you will encode a sequence using arithmetic coding and decode a different sequence using arithmetic coding.

The following probability model should be used for both encoding and decoding. Note that $\langle eot \rangle$ can only occur as the last character in the string.

| Symbol | Probability | Range |
|---|---|---|
| $a$ | 0.4 | [0.0, 0.4) |
| $b$ | 0.35 | [0.4, 0.75) |
| $c$ | 0.15 | [0.75, 0.9) |
| $\langle eot \rangle$ | 0.1 | [0.9, 1) |

(a) Encode $caab\langle eot \rangle$ using arithmetic coding. Use the smallest number of bits possible assuming that all bits to the right of the last transmitted bit are zeros.

(b) Decode 0011010111. Assume that all bits to the right of the last transmitted bit are zeros.

The MATLAB functions `binary2real`, `encode_symbol`, and `send_bit` discussed in lecture are available on the course web site. Note that the command `format long` will cause MATLAB to print out 14 decimal places instead of 4. You should feel free to enhance these routines to print out more information.

You will need to write your own function `decode_symbol`.

## Lectures 6E Lempel-Ziv Algorithms

11. (10 pts) *LZW.*

In this problem you will encode and decode sequences using the Lempel-Ziv-Welch algorithm LZW discussed in lecture.

The following initial phrasebook should be used for both encoding and decoding.

| Phrase Number | Phrase |
|---|---|
| 1 | a |
| 2 | b |
| 3 | c |

(a) Encode *cabcbcbcb* using LZW.

(b) Decode 3,4,5,6,7,8,9,1