EE 231A
Information Theory
Instructor: Rick Wesel

Handout #11, Solution Set 3
July 24, 2017

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
   01
   001
   001001
   001001001

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

   The code is not instantaneous because it is not prefix free. In fact every codeword except the first is constructed by concatenating all previous codewords. Since the codeword lengths satisfy the Kraft inequality, we can construct an instantaneous binary source code with these codeword lengths:

$$\begin{aligned}\sum D^{-l_i} &= 2^{-1} + 2^{-2} + 2^{-3} + 2^{-6} + 2^{-9} && (1)\\ &= \frac{256 + 128 + 64 + 8 + 1}{512} && (2)\\ &= \frac{457}{512} && (3)\\ &< 1\,. && (4)\end{aligned}$$

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

   The code is not uniquely decodeable since one cannot distinguish between, for example, the third codeword and the concatenation of the first two. Again, because the codeword lengths satisfy Kraft, we can certainly construct a uniquely decodeable binary source code with these codeword lengths.

2. (8 pts) *Reversal.*

   Consider a binary source code that employs the following codewords:
   { 0 01 011 0111 01111 011111 111111}.

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

   This code is certainly not instantaneous because it is not prefix-free. Each codeword is the prefix of all codewords below it except the last codeword. However, an instantaneous code does exist that satisfies these codeword lengths because that Kraft inequality is satisfied.

$$\sum D^{-l_i} = 2^{-1} + 2^{-2} + 2^{-3} + 2^{-4} + 2^{-5} + 2^{-6} + 2^{-6} \quad (5)$$
$$= 1\,. \quad (6)$$

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

   This code is uniquely decodeable. After the entire transmission is complete, the code can be decoded by decoding the letters in reverse order. In this way the code can be decoded as a regular Huffman code, since the codewords are the codewords of a Huffman code with their bits in reverse order.

3. (8 pts)

   Slackness in the Kraft inequality. Instantaneous codes are prefix free codes, i.e., no codeword is a prefix of any other codeword. Let $n_{max} = \max\{n_1, n_2, ..., n_q\}$. There are $D^{n_{max}}$ sequences of length n_{max}. Of these sequences, $D^{n_{max}-n_i}$ start with the i-th codeword. Because of the prefix condition no two sequences can start with the same codeword. Hence the total number of sequences which start with some codeword is $\sum_{i=1}^{q} D^{n_{max}-n_i} = D^{n_{max}} \sum_{i=1}^{q} D^{-n_i} < D^{n_{max}}$. Hence there are sequences which do not start with any codeword. These and all longer sequences with these length n_{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 n_i is D^{-n_i}, and $\sum D^{-n_i} < 1$, 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.

4. (8 pts) *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.

Lecture 6A: Huffman Coding.

5. (8 pts) *Huffman Code.*

X is distributed according to the following probability mass function:

$$\frac{1}{2} \quad \frac{1}{4} \quad \frac{1}{8} \quad \frac{1}{16} \quad \frac{1}{32} \quad \frac{1}{64} \quad \frac{1}{64}\,. \tag{7}$$

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

The Huffman tree is

Codeword								
0	x_1	$\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{2}$	1
10	x_2	$\frac{1}{4}$	$\frac{1}{4}$	$\frac{1}{4}$	$\frac{1}{4}$	$\frac{1}{4}$	$\frac{1}{2}$	
110	x_3	$\frac{1}{8}$	$\frac{1}{8}$	$\frac{1}{8}$	$\frac{1}{8}$	$\frac{1}{4}$		
1110	x_4	$\frac{1}{16}$	$\frac{1}{16}$	$\frac{1}{16}$	$\frac{1}{8}$			
11110	x_5	$\frac{1}{32}$	$\frac{1}{32}$	$\frac{1}{16}$				
111110	x_6	$\frac{1}{64}$	$\frac{1}{32}$					
111111	x_7	$\frac{1}{64}$						

This code has an expected length of $\frac{1}{2}+\frac{2}{4}+\frac{3}{8}+\frac{4}{16}+\frac{5}{32}=\frac{12}{64}=\frac{63}{32}$. $H(X)=-\sum p\log_2 p=\frac{63}{32}$. *The Huffman code exactly achieves the entropy because the probability distribution is dyadic.*

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

X is distributed according to the following probability mass function:

$$\frac{1}{3} \quad \frac{1}{3} \quad \frac{1}{9} \quad \frac{1}{9} \quad \frac{1}{27} \quad \frac{1}{27} \quad \frac{1}{27}\,. \tag{8}$$

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

Codeword								
0	x_1	$\frac{1}{3}$	$\frac{1}{3}$	$\frac{1}{3}$	$\frac{1}{3}$	$\frac{1}{3}$	$\frac{1}{3}$	1
10	x_2	$\frac{1}{3}$	$\frac{1}{3}$	$\frac{1}{3}$	$\frac{1}{3}$	$\frac{1}{3}$	$\frac{2}{3}$	
110	x_3	$\frac{1}{9}$	$\frac{1}{9}$	$\frac{1}{9}$	$\frac{1}{9}$	$\frac{1}{3}$		
1110	x_4	$\frac{1}{9}$	$\frac{1}{9}$	$\frac{1}{9}$	$\frac{2}{9}$			
11110	x_5	$\frac{1}{27}$	$\frac{1}{27}$	$\frac{1}{9}$				
111110	x_6	$\frac{1}{27}$	$\frac{2}{27}$					
111111	x_7	$\frac{1}{27}$						

This code has an expected length of $\frac{1}{3}+\frac{2}{3}+\frac{3}{9}+\frac{4}{9}+\frac{17}{27}=2\frac{11}{27}=2.41bits$. *This binary Huffman code does not achieve the entropy because the probability distribution is not dyadic. However, the Huffman code comes very close.*

(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.

Codeword					
0	x_1	$\frac{1}{3}$	$\frac{1}{3}$	$\frac{1}{3}$	1
1	x_2	$\frac{1}{3}$	$\frac{1}{3}$	$\frac{1}{3}$	
20	x_3	$\frac{1}{9}$	$\frac{1}{9}$	$\frac{1}{3}$	
21	x_4	$\frac{1}{9}$	$\frac{1}{9}$		
220	x_5	$\frac{1}{27}$	$\frac{1}{9}$		
221	x_6	$\frac{1}{27}$			
222	x_7	$\frac{1}{27}$			

This code has an expected length of $\frac{2}{3} + \frac{4}{9} + \frac{9}{27} = 1\frac{4}{9} = 1.44$ 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.

Yes, this ternary Huffman code *does* achieve the entropy because the probability distribution is triadic (all probabilities take the form $\frac{1}{3^n}$. The entropy is also $1\frac{4}{9}$ trits. Note that this is equal to 2.29 bits, which is less that the 2.41 bit average length of the binary Huffman code.

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?

Solution: Regardless of the PMF, there will be exactly the same tree diagram (within a permutation of the mass points at the left). Figure **??** shows this tree diagram. Thus one Huffman code (for example {0,10,11}) is sufficient to handle all three-element PMFs.

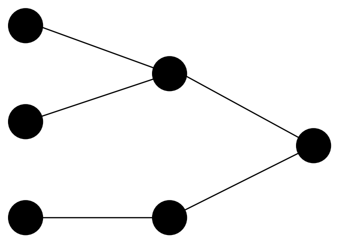

Figure 1: The single tree that handles all three-point PMFs.

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

Solution: Only two trees are needed to handle all four-point PMFs. Figure **??** shows these tree diagrams. Thus two Huffman codes (for example {0,10,110,111} and {00,01,10,11}) are sufficient to handle all four-element PMFs.

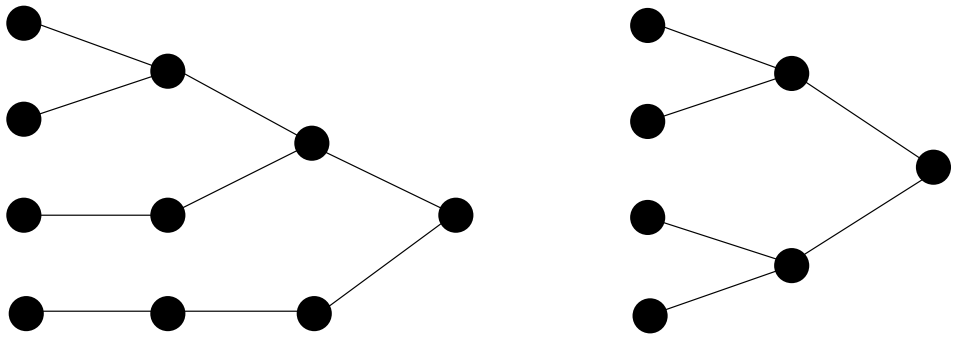

Figure 2: The two trees that handle all four-point PMFs.

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.

$$\begin{aligned} \sum D^{-l_i} &= 2^{-1} + 2^{-2} + 2^{-3} + 2^{-4} + 2^{-5} && (9) \\ &= 1 - 2^{-5} && (10) \\ &< 1 && (11) \end{aligned}$$

The codeword lengths satisfy the Kraft inequality. Thus a prefix-free code exists. This is a stronger condition than uniquely decodable, so certainly a uniquely decodable code exists.

The McMillan inequality may also be used to show unique decodability.

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

$$\{0,\ 10,\ 110,\ 1110,\ 11110\} \qquad (12)$$

(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.

These lengths cannot be the lengths of a Huffman code because the two longest codewords don't have the same length.

Lecture 6B: Optimality of Huffman Coding.

9. (9 pts) *Bad codes*

   (a) {0,10,11} is a Huffman code for the distribution (1/2,1/4,1/4).

   (b) The code {00,01,10, 110} can be shortened to {00,01,10, 11} without losing its instantaneous property, and therefore is not optimal, so it cannot be a Huffman code. Alternatively, it is not a Huffman code because there is a unique longest codeword.

   (c) The code {01,10} can be shortened to {0,1} without losing its instantaneous property, and therefore is not optimal and not a Huffman code.

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

10. (18 pts)

   (a) I modified the programs on the web page to print out the pertinent information for a terse solution. The actual source code is on the next page.

```
>> ll = [0.0 0.4 0.75 0.9];
>> ul = [0.4 0.75 0.9 1.0];
>>  [low, high] = encode_symbol (3, 0.0, 1.0, ll, ul);
 New interval is [0.750000, 0.900000).
>>  [low, high, bit] = send_bit(low, high);
Sent a 1.  New interval is [0.500000, 0.800000).
>>  [low, high] = encode_symbol (1, low, high, ll, ul);
 New interval is [0.500000, 0.620000).
>>  [low, high] = encode_symbol (1, low, high, ll, ul);
 New interval is [0.500000, 0.548000).
>>  [low, high] = encode_symbol (2, low, high, ll, ul);
 New interval is [0.519200, 0.536000).
>>  [low, high, bit] = send_bit(low, high);
Sent a 1.  New interval is [0.038400, 0.072000).
>>  [low, high, bit] = send_bit(low, high);
Sent a 0.  New interval is [0.076800, 0.144000).
>>  [low, high, bit] = send_bit(low, high);
Sent a 0.  New interval is [0.153600, 0.288000).
>>  [low, high, bit] = send_bit(low, high);
Sent a 0.  New interval is [0.307200, 0.576000).
>>  [low, high] = encode_symbol (4, low, high, ll, ul);
 New interval is [0.549120, 0.576000).
>>  [low, high, bit] = send_bit(low, high);
Sent a 1.  New interval is [0.098240, 0.152000).
>>  [low, high, bit] = send_bit(low, high);
Sent a 0.  New interval is [0.196480, 0.304000).
>>  [low, high, bit] = send_bit(low, high);
Sent a 0.  New interval is [0.392960, 0.608000).
```

   A final 1 is sent to place the value in the final interval.
   The encoded sequence is 110001001.

(b) The source code for the function decode_symbol is shown at the bottom of the page.

```
>> value = binary2real([0 0 1 1 0 1 0 1 1 1])
value =
   0.20996093750000
>> [symbol value] =  decode_symbol (value, ll, ul);
Symbol is 1.  New value is 0.524902.
>> [symbol value] =  decode_symbol (value, ll, ul);
Symbol is 2.  New value is 0.356864.
>> [symbol value] =  decode_symbol (value, ll, ul);
Symbol is 1.  New value is 0.892160.
>> [symbol value] =  decode_symbol (value, ll, ul);
Symbol is 3.  New value is 0.947731.
>> [symbol value] =  decode_symbol (value, ll, ul);
Symbol is 4.  New value is 0.477307.
```

Thus the decoded sequence is $abac\langle eot\rangle$.

-
  ```
  function [new_low,new_high] = encode_symbol (symbol, low, high, ll, ul)
  range = high - low;
  new_low   = low + ll(symbol) * range;
  new_high  = low + ul(symbol) * range;
  fprintf(1, ' New interval is [%f, %f).\n', new_low, new_high);
  ```

-
  ```
  function [new_low,new_high,bit] = send_bit(low, high)

  bit = -1;
  new_low = low;
  new_high = high;

  if (high < 0.5)
     bit = 0;
  elseif (low > 0.5)
     bit = 1;
     new_low = new_low - 0.5;
     new_high = new_high - 0.5;
  end

  if (bit > -1)
     new_low = new_low * 2;
     new_high = new_high * 2;
     fprintf(1, 'Sent a %d.  New interval is [%f, %f).\n', bit, new_low, new_high);
  end
  ```

-
  ```
  function [symbol, new_value] = decode_symbol(value, ll, ul)

  symbol = 1;
  while (value > ul(symbol))
     symbol = symbol + 1;
  end

  new_value = (value - ll(symbol)) / (ul(symbol) - ll(symbol));
  ```

```
fprintf(1, 'Symbol is %d.  New value is %f.\n', symbol, new_value);
```

Lectures 6E Lempel-Ziv Algorithms

11. (10 pts)

(a)
- Encode c as 3. Add ca as phrase 4.
- Encode a as 1. Add ab as phrase 5.
- Encode b as 2. Add bc as phrase 6.
- Encode c as 3. Add cb as phrase 7.
- Encode bc as 6. Add bcb as phrase 8.
- Encode bcb as 8.

Encoded sequence is 3,1,2,3,6,8.

(b)
- Decode 3 as c. Phrase 4 is c and a final letter.
- Since the incomplete phrase 4 is the next phrase to decode, the first and last letters of phrase 4 must be the same. Decode 4 as cc. Phrase 5 is cc and a final letter.
- As above, decode 5 as ccc. Phrase 6 is ccc and a final letter.
- Decode 6 as $cccc$. Phrase 7 is $cccc$ and a final letter.
- Decode 7 as $ccccc$. Phrase 8 is $ccccc$ and a final letter.
- Decode 8 as $cccccc$. Phrase 9 is $cccccc$ and a final letter.
- Decode 9 as $ccccccc$. Phrase 10 is $ccccccc$ and a final letter.
- Decode 1 as a.

The decoded sequence is $c, cc, ccc, cccc, ccccc, cccccc, ccccccc, a$.
(The commas are shown only for clarity.)