CSC310 – Assignment #1

Due: Sept. 27, 2006, 10am at the **START** of class
Worth: 8%
Late assignments not accepted.

1 Codes

- For each of the following codes, state whether it satisfies the Kraft-McMillan inequality, whether it is uniquely decodable and also whether it is instantaneously decodable.
  (a) {000,001,11,10,01}
  (b) {0101,110,001,11,00}
  (c) {011,11,1011,110,0100,0110}
  (d) {0,0010,000100}
  (e) {11,10,01,0}
  (f) {1,00,10}
- For codes which fail to be uniquely decodable, give an example of a string of bits that can be decoded in more than one way.
- For codes which are uniquely decodable but fail to be instantaneously decodable, give an example of a string of bits (corresponding to the encoding of more than one symbol) in which the first symbol *can* be successfully decoded but only with some nonzero delay.
- For codes which are both uniquely and instantaneously decodable, draw a binary tree which has the codewords as the leaves.

2 Optimal Codes

- Consider a source which produces symbols X (independently) from the alphabet $\{a_1, a_2, a_3\}$ with probabilities $\{p(a_1) = 0.6, p(a_2) = 0.1, p(a_3) = 0.3\}$.
- Compute the entropy (in bits) of this source X and of the second extension of this source X^2 (i.e. when we encode pairs of symbols at a time).
- Find a Huffman Code for X. Do the same for the second extension X^2.
- Calculate the average codeword length *per symbol* in each case.
- Consider a source which produces symbols X (independently) from the alphabet $\{a_1, a_2, a_3, a_4, a_5, a_6\}$ with probabilities $\{p(a_1) = 0.10, p(a_2) = 0.11, p(a_3) = 0.14, p(a_4) = 0.14, p(a_5) = 0.23, p(a_6) = 0.28\}$.
- Find a Huffman Code for X.
- Find an optimal instantaneous code for this source that is *not* a Huffman Code. That is, your code should be different than any Huffman Code that could be obtained by changing the way arbitrary choices are made in the Huffman code algorithm. Hint: consider symbols which have codewords of equal length in a Huffman code but which can never be merged.
- (Harder) By using simple Shannon-Fano codes, give an upper bound on the block length (i.e. what extension or what number of symbols at a time we would need to code) required to make the average per symbol codeword length within 1% of the source entropy.

3 Puzzles

There is a strong link between coding/compression and games or puzzles which involve asking a series of questions in order to determine the identity of some unknown object or quantity. We think of the unknown to be determined as the symbol X which is being coded and we think of the series of answers you receive as the encoding Z. (For example, consider the well known game in which you are given a set of K identical looking balls, one of which is heavier than the others. You have at your disposal a balance, on which you can place any combination of balls and which will report to you either that the left set is heavier (L); the right set is heavier (R); or the two are balanced (B). The symbol X can take on a value in $\{1, \ldots, K\}$ representing which of the balls is the heavy one and the elements of Z take on values in $\{L, R, B\}$.)
See the beginning of Chapter4 in Mackay's book for more on this.

- Consider the game in which two 6-sided dice are rolled and you want to determine their sum X by asking a series of yes/no questions.
- Map this game onto a coding problem by specifying the alphabet of X, the associated source model $p(X)$ and the encoding alphabet of Z.
- Derive in detail a strategy that is guaranteed to succeed and *minimizes the maximum number of questions* required.
- Your strategy should correspond to an instantaneous code, and thus you will be able to describe it using a tree. Explicitly give the binary tree corresponding to your strategy, where at each internal node you clearly state the query to be asked (e.g. "Is the sum greater than 4?" or "Is the sum equal to either 8 or to 11?") and clearly state which branch is to be taken if the answer is yes and which branch if the answer is no. At the leaves of the tree, clearly state what value of X is to be returned (e.g. "return X=2").
- Do the same as above, but derive a strategy that is guaranteed to succeed and *minimizes the expected number of questions required*, assuming the dice are fair (uniform probability of getting each number between 1 and 6 on each, independently of each other). Again, explicitly give the tree, the queries and the leaves.