Department of Electrical Engineering, IIT Madras

Information Theory Project Report

Sourbh Bhadane, EE12B109

May 1, 2015

1 Data Compression

The submitted code contains the functions *huffman_ encode*, *recu*, *recu_ assign*, *generate _data huffman_ decode*. *recu* and *recu_ assign* are auxiliary functions to *huffman_ encode*. *generate_ data* is used to generate a string of random data given the source code from the source encoder and length of the random string. A provision for creating incorrect data is also provided in *generate _ data*
Usage of the python script : python version1.py _probability distribution_ _size of encoded alphabet _ _length of random string _

1.1 Source Encoder

A recursive implmentation of the Huffman Encoder has been used. At first, the probability distribution is sorted and at every stage of the recursion, a list of indices is created which get grouped at that stage. Thus a series of nested lists is the output of the Huffman encoder. This series of nested lists is assigned codewords by *recu_ assign*
Sample Input : 0.2, 0.24, 0.16, 0.1, 0.12, 0.18 4
Sample Output : 1 2 33 31 32 0

1.2 Source Decoder

For the source decoder, the incoming data stream is processed bit by bit. A codeword is built as the bits are collected. As soon as a codeword is formed, the corresponding source symbol is appended to the output. Thus, the source decoding is instantaneous.

Sample Input : 1130313233323033223233132301303132 21 (with the same lookup table as generated by the above sample input)
Sample Output : [0, 0, 6, 3, 4, 2, 4, 6, 2, 1, 1, 4, 2, 0, 4, 6, 0, 6, 3, 4, 1, 0]

2 Channel Capacity

2.1 Channel Capacity Probability Transition Matrix

$$
\begin{aligned}
C &= \max_{p(x)} I(X;Y) \\
C &= \max_{p(x)} H(Y) - H(Y|X) \\
H(Y|X) &= H(row) \\
C &= \max_{p(x)} H(Y) - H(row) \\
H(Y) &= -\sum\nolimits_y p(y)\log(p(y)) \\
H(Y) &= -\sum\nolimits_y \left[\sum\nolimits_x p(y|x)p(x)\right]\log(p(y) \\
&= -\sum\nolimits_x \left[\sum\nolimits_y p(y|x)p(x)\right]\log(p(y) \\
&= -\sum\nolimits_x p(x)\left[\sum\nolimits_y p(y|x)\log(p(y)\right]
\end{aligned}
$$

Our aim is to maximize the last equation given the constraint $\sum_x p(x) = 1$. This constrained maximization can be done using Lagrange Multipliers. The variables are the elements in X.

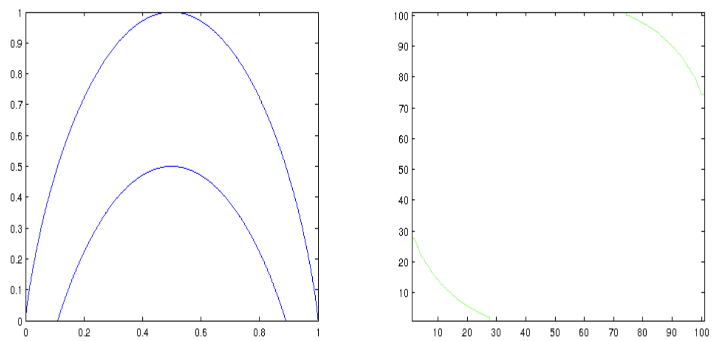
$$L = -\sum\nolimits_x p(x)f(x) - \lambda(\sum\nolimits_x p(x) - 1)$$

2.2 Channel Capacity 0.5

For the binary input output DMC, let $p(x=0)=\lambda, p(y=0|x=1)=q, p(y=1|x=0)=p$

$$
\begin{aligned}
0.5 &= \max_{p(x)} I(X;Y) \\
&= \max_{p(x)} H(Y) - H(Y|X) \\
&= \max_{\lambda} H((1-\lambda)q + \lambda(1-p)) - \lambda H(p) - (1-\lambda)H(q)
\end{aligned}
$$

Approach 1 Geometric :
A geometric interpretation of the above equation implies that given the graph of $H(p)$ versus p, consider two points on the graph corresponding to $H(p)$ and $H(1-q)$ and some λ. The second term in the above equation corresponds to some point on the line joining $H(p)$ and $H(1-q)$. The first term is the point on the graph just above the latter point. Thus, effectively the problem is reduced to finding p and q such that the vertical distance between the above two points is at the maximum 0.5.
Consider the following construction, we plot $H(p)-0.5$ versus p wherever it is positive. Choose any point on this plot, and draw a tangent to this curve. Let it intersect the $H(p)$ versus p plot at two distinct points. These points are the required p and $1-q$.



(a) Geometric

(b) Graphical

Approach 2 Graphical:

λ is varied and the maximum value is calculated for different values of p and q. This gives the following contour plot. We can vary the capacity also