Digital Communication – CA#1

Sahand Khoshdel – 810196607

Computer Assignment #1

Table of Contents

Page #

Problem 9 2
Problem 10 3
Problem 11 4
Problem 12 6
Problem 13 8

Q9) Entropy Estimation for Stationary Markov Source

Step 1: Solving system of equations to find state probability vector (Assumption: p_M)

We have M unknowns, so we need M independent equations:

$$\boldsymbol{p\varphi^t = p} \; \rightarrow \boldsymbol{p} = ?$$

$$\left.\begin{aligned} &\Rightarrow \boldsymbol{p(\varphi^t - I) = 0} \; (\boldsymbol{contains\ M-1\ independent\ equations}) \\ &\sum_{i=1}^{M} \boldsymbol{p_i = 1} \; (\boldsymbol{Axiom\ \#3 - Other\ independent\ equation}) \end{aligned}\right]$$

$$\Rightarrow \boldsymbol{Solving\ System\ of\ equations} \Rightarrow p_M \checkmark$$

Step 2: Calculating state entropy (H_s)

$$H_{s_1} = -\sum_{i=1}^{M} \Pr\{s = s_i\} \left(log_2^{\Pr\{s=s_i\}}\right) = -\sum_{i=1}^{M} p_i \left(log_2^{p_i}\right)$$

This can simply be done in a for loop over the M states.

Step 3: Calculating state transition entropy ($H_{s_1 s_2} = H_{s_2|s_1}$)

$$H_{s_1 s_2} = -\sum_{i=1}^{M} \Pr\{s_1 = s_i\} \sum_{j=1}^{M} \Pr\{s_2 = s_j | \, s_1 = s_i\} \; log_2^{\Pr\{s_2=s_j | \, s_1=s_i\}}$$

$$= -\sum_{i=1}^{M} p_i \sum_{j=1}^{M} \varphi_{ij} \; log_2^{\varphi_{ij}}$$

This can also be done in 2 nested loops.

Step 4: Calculating joint entropy of super-symbols (Using chain rule and Markovity)

$$\boldsymbol{G_k} = \frac{H(X_1 \ldots X_k)}{k} \Rightarrow \frac{H_{s_1} + H_{s_1 s_2} + \cdots + \; H_{s_k s_{k-1}}}{k}$$

$$= \frac{\boldsymbol{H_{s_1} + k(H_{s_1 s_2})}}{\boldsymbol{k}}$$

Q10) Average Huffman Code Length

Step 1: Dividing Main Chain into groups with length = k

First thing to do is to consider that the chain's length may not be divisible by k, so we either have to remove the extra symbols or pad zero symbols to the end, which padding may cause inefficient results, so I removed the extra bits to make the main chain divisible by k.

Step 2: Assigning Super-symbols to each group and calculating probability (using the frequency definition)

In this step we have to save unique symbols that exist in the chain in a list and count there occurrence over a loop on all the groups and simply divide it by the total number of symbols(groups) to obtain probability.

Step 3: Using 'huffmandict' function to calculate average code length(per super-symbol) for the main chain:

By feeding the calculated probabilities to the 'huffmandict 'function we obtain the average code length per super-symbol and we can obtain $\widehat{H}_k$ as well by simply dividing it to the group length (k).

We will see the convergence of $\widehat{H}_n$ towards the entropy in the next problem (simulation part)

Q11) Entropy Estimation by Simulating a Markov Source

A) Entropy Estimation by Gk (Simulation)

If we plot Gk in a range of k = [1:10], we expect it to approach entropy as the following equation describes an asymptotic behavior:

$$\boldsymbol{H}(\boldsymbol{X}) = \lim_{k\to\infty} \boldsymbol{G_k}$$

In our case we can determine the convergence rate easily:

$$\boldsymbol{G_k} = \frac{H_{s_1} + k(H_{s_1 s_2})}{k} = \boldsymbol{H_{s_1 s_2}} + \frac{\boldsymbol{H_{s_1}}}{\boldsymbol{k}}$$

The entropy can be easily calculated:

$$\boldsymbol{G_1} = \boldsymbol{H_{s_1 s_2}} + \boldsymbol{H_{s_1}}$$

$$\boldsymbol{H}(\boldsymbol{X}) = \boldsymbol{G_\infty} = \boldsymbol{H_{s_1 s_2}} = 0.8930 \text{ (in our case)}$$

Therefore, we conclude:

“The Entropy of a Stationary Markov Source is independent from the Entropy of the states themselves (H_{s_1})”

The role of initial probabilities is in how they affect Transition (Conditional Entropy), if they balance towards a uniform distribution, they will maximize the Source’s Entropy.

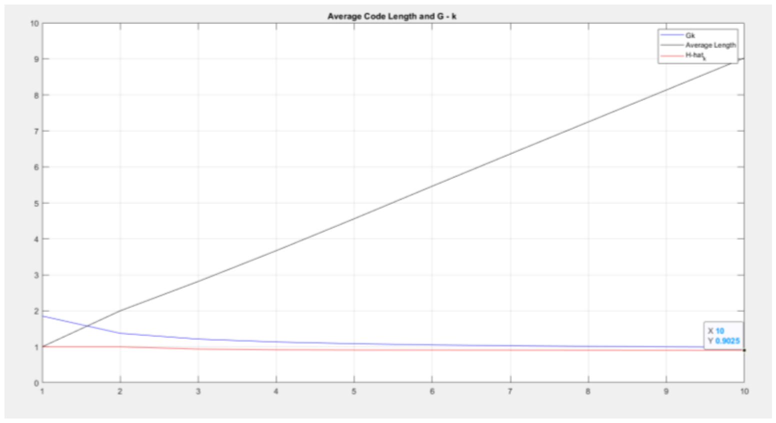
Thus, Gk will decrease asymptotically in a homographic manner:

The following expression can **approximate convergence rate** (derivative-wise) as if Gk was a continuous function:

$$\frac{\boldsymbol{\Delta G_k}}{\boldsymbol{k}} = \frac{G_{k+1} - G_k}{k} = \frac{\frac{H_{s_1}}{k+1} - \frac{H_{s_1}}{k}}{k} = \frac{\boldsymbol{-H_{s_1}}}{\boldsymbol{k+1}}$$

B) Entropy Estimation by average length and $\widehat{H}_n$ (Simulation)

As we can see in the figure below, the Average length of code approaches a linear behavior as k grows and this is because the redundancy caused by choosing small k’s in grouping symbols into super-symbols, decreases as we increase k. Thus, the Estimated Entropy should also decrease that we can see in the figure below.



Estimations using Gk & Average length (Huffman Coding) Converge to H(X) for a sufficiently large k.

C) Coding Efficiency

Coding efficiency can be obtained by dividing the entropy $H(X) = H_{s_1 s_2}$ by $\widehat{H}_k = \frac{\bar{l}}{k}$, as the following equation shows:

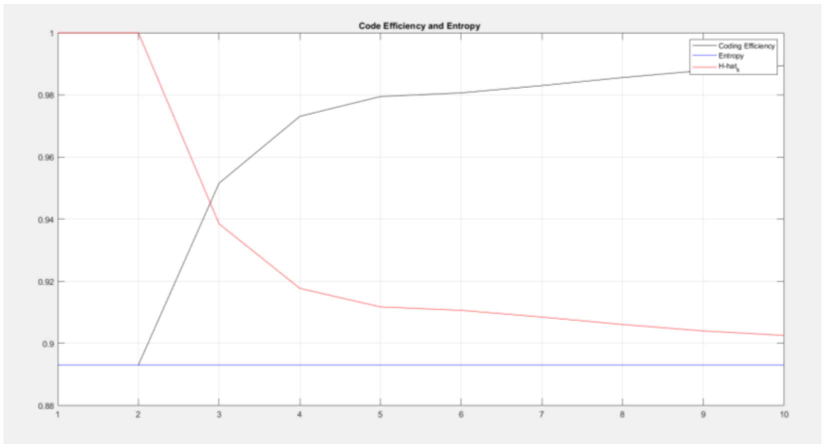
$$\eta_k \triangleq \frac{H(X)}{\widehat{H}_k}$$

We expect code efficiency (η_k) to approach 1 as 'k' becomes larger:

$$\lim_{k\to\infty} \widehat{H}_k = H(X)$$

$$\Rightarrow \lim_{k\to\infty} \eta_k = 1$$

Simulation Results confirm this theoretical belief as well:



Q12) Entropy Estimation by Simulating a Memory-less Source

We can model **a Memory-less source as a Markov source that only has one state and returns to it in each iteration but has different probabilities for producing symbols.**

Thus the previous formulas are still correct, we just need to shrink every thing into a single state:

If we plot Gk in a range of k = [1:10], we expect it to approach entropy as the following equation describes an asymptotic behavior:

$$\boldsymbol{H}(\boldsymbol{X}) = \lim_{k\to\infty} \boldsymbol{G_k}$$

In our case we can determine the convergence rate easily:

$$\boldsymbol{G_k} = \frac{H_{s_1} + k(H_{s_1})}{k} = \left(\frac{\boldsymbol{k+1}}{\boldsymbol{k}}\right)\boldsymbol{H_{s_1}} = \left(\frac{\boldsymbol{k+1}}{\boldsymbol{k}}\right)\boldsymbol{H}(\boldsymbol{X})$$

The entropy can be easily calculated:

$$\boldsymbol{G_1} = \boldsymbol{2H}(\boldsymbol{X})$$

$$\boldsymbol{H}(\boldsymbol{X}) = \boldsymbol{G}_\infty$$

Entropy can be similarly calculated for combined sources (H(X1X2), …)

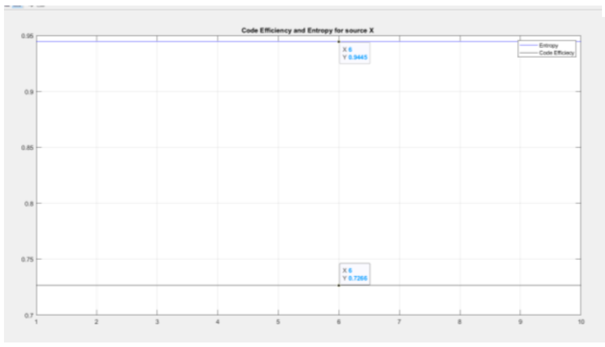
- Also, as each symbol doesn't provide any information for the future ones, Entropy can be directly obtained for a memoryless complex source:

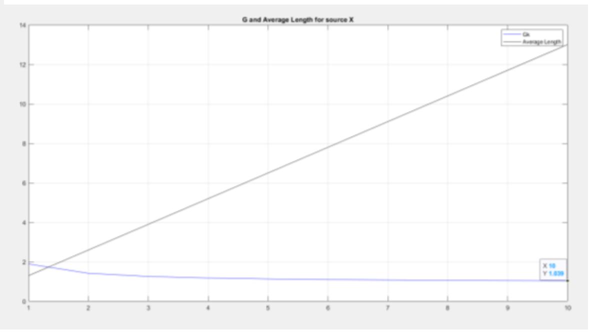
$$\mathbf{H(X)} = -\sum_{i=1}^{M} p_i\left(log_2^{p_i}\right) = 0.9445$$

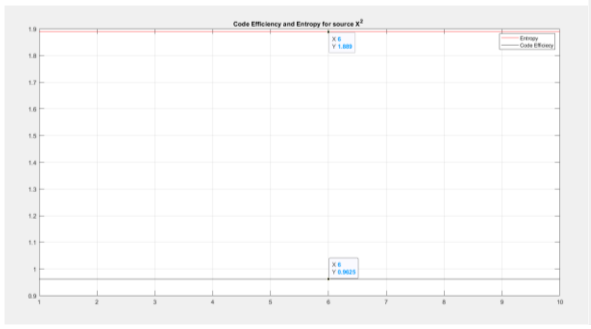
$$H(X_1X_2) = 2H(X) = 1.889$$

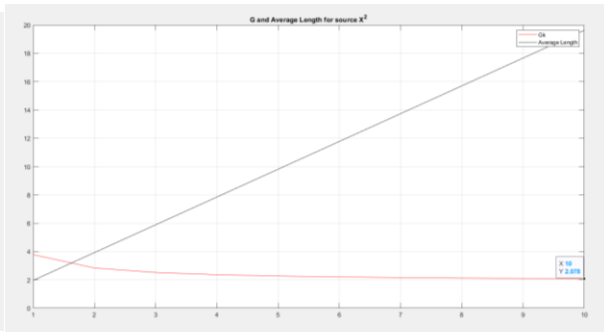
$$H(X_1X_2X_3) = 3H(X) = 2.8335$$

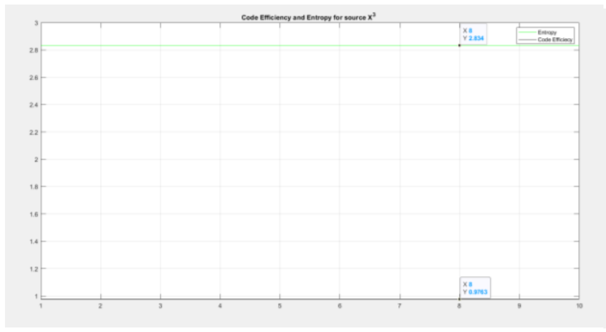
They can also be directly calculated from obtaining joint probabilities which I did first but realized the previous way is much faster. But we need joint probabilities for Huffman coding, that's easy to obtain using Px.

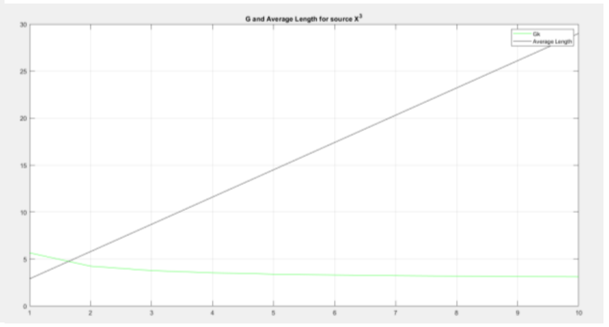












Average length for Huffman code grows completely linear as we increase k in memory-less sources because probability of producing a symbol doesn't depend on previous symbols, **thus no redundancy is defined. And average length is 100% correlated with symbol numbers in a group**

Thus, Coding efficiency will also remain constant as 'k' grows due to linear average length Huffman code **for a specific source.**

Clearly average length for complex sources is higher than simple source (but not linearly), **so we expect constant lines for each source but X^3 should be quite close to 1**

$$\boldsymbol{\eta_k(X)} \triangleq \frac{H(X)}{\widehat{H}_k} = \frac{H(X)}{\bar{l}}(k) = c.s.t = \frac{0.9445}{1.3} = \mathbf{72.65\%}$$

$$\boldsymbol{\eta_k(X^2)} \triangleq \frac{H(X^2)}{\widehat{H}_k} = \frac{H(X^2)}{\bar{l}}(k) = c.s.t = \frac{1.8891}{1.9627} = \mathbf{96.25\%}$$

$$\boldsymbol{\eta_k(X)} \triangleq \frac{H(X)}{\widehat{H}_k} = \frac{H(X)}{\bar{l}}(k) = c.s.t = \frac{2.8336}{2.9025} = \mathbf{97.63\%}$$

These values are exactly obtained and can be seen in the graphs above.

Q13) Markov V.s Memoryless Sources – Parameter comparison

- As we observed the main difference between Markov sources and memory-less sources is that **memory-less sources are not affected by the amount of k**, because **no relative redundancy caused by conditional probability is defined. Code Efficiency remains constant** and **Average code length is an exact line**.

  But in **Markov sources** because of **decreasing redundancy of less-likely outcomes by grouping symbols into super-symbols, average code length tends towards a line asymptotically.** and **code efficiency approaches 1 as 'k' grows.**

- However, we can **increase** the **code efficiency** of **Memory-less sources**, by combining them.

- **Memory-less** sources have a noticeable **higher entropy** (0.95> 0.89)because of **the higher ambiguity** the symbols have and **the lack of information provided by each other** in their generation.