Channel Coding

Channel coding refers to the class of signal transformations designed to improve communications performance by enabling the transmitted signals to better withstand the effects of various channel impairments, such as noise, interference and fading.

Broadly, channel coding can be classified into 2 study areas, (a) Waveform coding (b) Structured sequences.

Waveform coding deals with transforming waveforms into "better waveforms" to make the detection less subject to errors.

Structured sequences deals with transforming data sequences into "better sequences" having structured redundancies (redundancy bits). The redundant bits can then be used for the detection and correction of errors.

Waveform coding involves coding the existing signal waveform to form orthogonal or bi-orthogonal systems. For an M-ary signal, the symbol error probability of orthogonal signals is given by

$$P_E(M) \leq (M-1)\, Q\left(\sqrt{\frac{E_s}{N_0}}\right)$$

Here $M = 2^k$ (k = no. of bits, M = possible number of codewords)

$$\Rightarrow E_s = kE_b$$

Also $$\frac{P_B(M)}{P_E(M)} = \frac{2^{k-1}}{2^k - 1} = \frac{M/2}{(M-1)}$$ (for orthogonal sequences)

$$\Rightarrow \boxed{P_B(M) \leq \frac{M}{2} Q\left(\sqrt{\frac{kE_b}{N_0}}\right)} \quad (114)$$

* Here $k=1$ i.e. $M=2$ gives the case of BPSK

It may be important to note here that the M-ary ~~antipodal~~ signalling of PSK, i.e. non-orthogonal signals,

$$P_E(M) \approx 2Q\left(\sqrt{\frac{2E_s}{N_0}} \sin\frac{\pi}{M}\right)$$

where $E_s = kE_b$

& $P_B(M) = \frac{P_E(M)}{\log_2 M} = \frac{P_E(M)}{k}$

Thus in both cases it can be observed that bit error probability can be reduced by increasing M (or k). But as we know, this increases the effective bandwidth being used.

Thus, another system of ~~en~~ encoding i.e. Structured sequences can be used by abandoning the need of orthogonal or antipodal signalling.

* Bi-orthogonal scheme is one in which out of M codewords, $\frac{M}{2}$ are orthogonal and the rest are antipodal to each of these signals.

$$P_E(M) \leq (M-2) Q\left(\sqrt{\frac{E_s}{N_0}}\right) + Q\left(\sqrt{\frac{2E_s}{N_0}}\right)$$

They are better as they have better P_E with $\frac{1}{2}$ BW requirement

Structured sequences are classified into 3 subcategories: block, convolution and turbo. Before the discussion on these begins we need to understand the concept of channel models. (115)

Memoryless Channel

Recall the fundamental of a memoryless system which states that the output of the system depends only on the current input of the system and not on past or ~~pres~~ future values. Similarly memoryless channel is one in which every output sequence, say, $Z = z_1, z_2, \ldots z_N$ depends only on its corresponding input sequence say $U = u_1, u_2, \ldots, u_N$. In such a case the conditional probability of receiving Z when U is transmitted is given as

$$P(Z|U) = \prod_{m=1}^{N} P(z_m|u_m)$$

i.e. the product of the individual element probabilities as each case is independent of all others.

In case the channel has memory (noise or fading occurs in bursts), the conditional probability would need to be expressed by joint probability of all elements of the sequence.

A set of all possible codeword sequences is commonly called an alphabet.

When we use a discrete alphabet set, then the memoryless channel is called a Discrete Memoryless channel (DMC). A special case is a Binary Symmetric channel (BSC)

Binary Symmetric Channel

Input and Output alphabets consist of only the binary elements i.e. 0 and 1. Also, the conditional probabilities are symmetric

i.e. $P(1|0) = P(0|1) = p$

and $P(1|1) = P(0|0) = 1-p$

These constitute the channel transition probabilities.

Here p is the probability of error and $(1-p)$ is probability of correct reception.

Needless to say, p is a function of symbol energy. For eg. in case of BPSK,

$$p = Q\left(\sqrt{\frac{2E_b}{N_0}}\right)$$

Since the demodulator output consists of discrete elements 0 and 1, the demodulator is said to make a hard decision on each symbol. We can also say that decoding using a BSC is a hard decision decoding.

We can generalize the discussion of a memoryless channel to one which does not have a discrete alphabet. One such example is a Gaussian channel.

Gaussian Channel

Gaussian channel has a discrete input alphabet but a continuous output alphabet over the range $(-\infty, \infty)$. The channel adds Gaussian noise to the symbols and thus the probability density function (pdf) can be given as

$$p(z|u_k) = \frac{1}{\sigma\sqrt{2\pi}} e^{\frac{-(z-u_k)^2}{2\sigma^2}} \quad \text{for all } z,$$

where σ^2 is the noise variance.

$k = 1, 2, \ldots, M$.

Again being a memoryless system, the net conditional probability has the same definition.

When the demodulator output consists of a continuous alphabet or its quantized approximation (with greater than two quantization levels), then the demodulator is said to make soft decisions and a decoder working on this principle is called a soft decision decoder.

Block codes generally use hard decision modelling whereas convolution codes may use either hard or soft decision decoding.

Important definitions

Code rate and Redundancy

In case of block codes, the source data is segmented into blocks of k data bits, also called, information bits or message bits; each block can represent any one of the 2^k distinct messages. The encoder transforms each k-bit data block into a larger block of n bits, called code bits or channel symbols. The (n−k) bits that the encoder adds to each data block, are called redundant bits, or parity bits or check bits. *.

The ratio of the redundant bits to data bits is called the redundancy of the code.

i.e. $\text{Redundancy} = \frac{n-k}{k}$

The ratio of the data bits to the total bits is called the code rate.

i.e. $\text{Code rate} = \frac{k}{n}$

Code rate can be visualised as the amount of information carried by each code bit. for eg. if code rate = $\frac{1}{2}$ then each code bit contains $\frac{1}{2}$ bit of information.

Relation between redundancy and code rate

$$\text{Redundancy } R = \frac{1-r}{r}$$

where r = code rate.

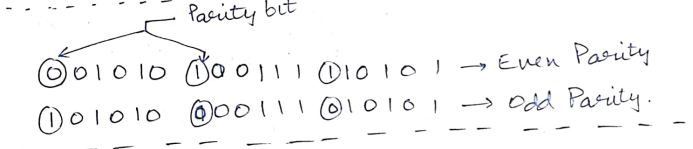
* The code in general is written as (n,k) code.

Parity Check Codes

Parity check codes use the linear sum of the information bits for error detection or correction.

A Single Parity Check Code is constructed by adding a single parity bit to a block of data bits. The parity bit takes on the value 1 or 0 as needed to ensure that the summation of all the bits in the codeword yields an even (or odd) result. The summation is done using modulo-2 addition (XORing). If the added parity is designed to yield an even result, the method is termed even parity; if it is designed to yield an odd result, the method is termed odd-parity.

Consider the following example.



At the receiving terminal the decoding procedure consists of testing that the Mod-2 sum of the codeword bits yields a zero result (in case of even parity; it would yield a one in case of odd parity). If the output is found to be 1 instead of 0, the codeword is said to contain errors.

It must be noted that this system can only detect an error and not correct. Furthermore, only odd no. of errors can be detected. In case even number of bits are inverted, then the parity check would appear correct. This would mean undetected errors

Assuming that all bit errors are equally likely and occur independently, we can write the probability of j errors occuring in a block of n symbols as

$$P(j,n) = {}^nC_j \, p^j (1-p)^{n-j} \qquad (120)$$

where p is the probability of receiving a bit with error and

$${}^nC_j = \frac{n!}{j!(n-j)!}$$

is the number of various ways in which j bits out of n may have an error.

Thus for a single parity check code, the probability of an undetected error, say P_{nd} with a block of n bits can be computed as

$$P_{nd} = \sum_{j=1}^{N} {}^nC_{2j} \, p^{2j} (1-p)^{n-2j}$$

where $N = \frac{n}{2}$ for even n

$= \frac{n-1}{2}$ for odd n

* Code rate $= \frac{k}{k+1}$ Redundancy $= \frac{1}{k}$

Linear Block Codes

This is another class of parity check codes characterised by the $\langle n,k \rangle$ notation. The encoder transforms a block of k message digits (a message vector) into a longer block of n codeword digits (a codevector) constructed from a given alphabet of elements. When the alphabet consists of 2 elements (0 & 1), the code is a binary code comprising of binary digits. The k message bits form 2^k distinct message sequences refered to as k-tuples (sequence of k digits). The n bit block can have as many as 2^n distinct sequences refered to as n-tuples. The encoding procedure assigns to each of the 2^k message k-tuples, one of the 2^n n-tuples. A block code thus represents one on one unique assignment of message blocks to codewords.

Some definitions

(a) Vector space

The set of all binary n-tuples is called the vector space over the binary field of 2 elements. It is termed as V_n.

(b) Vector Subspace

A subset S of the vector space V_n is called a subspace if the following 2 conditions are met.

1. The all zero vector is in S.
2. The sum of any two vectors in S is also in S. (closure property)

For example

$V_4 = \{0000, 0001, 0010, 0011, 0100, 0101, 0110, 0111, 1000, 1001, 1010, 1011, 1100, 1101, 1110, 1111\}$

Then an example of subset may be

$S = \{0000, 0101, 1010, 1111\}$

A set of 2^k n-tuples is called a linear block code if and only if it is a subspace of the vector space V_n of all n-tuples.

The basic goal in choosing a particular code can be stated as:

1. We strive for coding efficiency by packing the V_n space with as many codewords as possible. This means we want to expand a small amount of redundancy.
2. We want the codewords to be as far apart from one another as possible, so that even if the vectors experience some corruption during transmission, they may still be correctly decoded with a high probability.

Linear block code example

Suppose we want to design a (6,3) code. There are $2^3 = 8$ message vectors & therefore 8 codewords. There are $2^6 = 64$, 6-tuples in the V_6 vector space

Choosing a set of 8 codewords from the vector space, to form a subspace, we may write

Message Vector	Codeword
000	000000
001	101001
010	011010
011	110011
100	110100
101	011101
110	101110
111	000111

Generator Matrix

A large look up table is prohibitive. (for large values of n & k). It is always possible to find a set of n-tuples, less than 2^k that can generate all the 2^k codewords of the subspace. The smallest linearly independent set that spans the subspace is called a basis of the subspace and the number of vectors in this basis set is the dimension of the subspace. Any basis set of k linearly independent n-tuples $V_1, V_2, \ldots V_k$ can be used to generate a linear block code vectors. That is, each set of 2^k codewords [U] can be described by

$$U = m_1V_1 + m_2V_2 + \ldots + m_kV_k$$

where $m_i = 0$ or 1 are the message bits.
$i = 1 \rightarrow k$.

In general this may be written in the form of a generator matrix by the following $k \times n$ array (124)

$$G = \begin{bmatrix} V_1 \\ V_2 \\ \vdots \\ V_k \end{bmatrix} = \begin{bmatrix} V_{11} & V_{12} & \cdots & V_{1n} \\ V_{21} & V_{22} & \cdots & V_{2n} \\ \vdots & & & \\ V_{k1} & V_{k2} & \cdots & V_{kn} \end{bmatrix}$$

and the codeword U_i for any given message m_i would be given by

$$U_i = m_i G = [m_1, m_2 \ldots m_k] \begin{bmatrix} V_{11} & V_{12} & \cdots & V_{1n} \\ V_{21} & V_{22} & \cdots & V_{2n} \\ \vdots & & & \\ V_{k1} & V_{k2} & \cdots & V_{kn} \end{bmatrix}$$

Systematic linear Block Codes

A systematic (n,k) linear block code is a mapping from a k-dimentional message vector to an n-dimentional codeword in such a way that part of the sequence generated coincides with the k message bits. The remaining $(n-k)$ bits are parity bits. The systematic linear block code will have a generator matrix of the form

$$G = \left[P_{n-k} \vdots I_k \right]$$

$$= \left[\begin{array}{cccc:cccc} P_{11} & P_{12} & \cdots & P_{1(n-k)} & 1 & 0 & \cdots & 0 \\ P_{21} & P_{22} & \cdots & P_{2(n-k)} & 0 & 1 & \cdots & 0 \\ \vdots & & & & \vdots & & & \vdots \\ P_{k1} & P_{k2} & \cdots & P_{k(n-k)} & 0 & 0 & \cdots & 1 \end{array}\right]$$

(125)

$$\therefore \quad U_i = m_i G$$

$$= [m_1, m_2 \ldots m_k] \left[\begin{array}{cccc:cccc} P_{11} & P_{12} & \cdots & P_{1(n-k)} & 1 & 0 & \cdots & 0 \\ P_{21} & P_{22} & \cdots & P_{2(n-k)} & 0 & 1 & \cdots & 0 \\ \vdots & & & & \vdots & & & \vdots \\ P_{k1} & P_{k2} & \cdots & P_{k(n-k)} & 0 & 0 & \cdots & 1 \end{array}\right]$$

$$= \underbrace{P_1 P_2 P_3 \ldots P_{n-k}}_{n-k \text{ parity bits}}, \underbrace{m_1, m_2 \ldots m_k}_{k \text{ message bits}}$$

where $P_i = \sum_{z=1}^{k} m_z P_{zi} \qquad i = 1, 2, \ldots, n-k$

Going back to the (6,3) code example, the systematic codeword can be generated by

$$G = \left[\begin{array}{ccc:ccc} 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 & 1 \end{array}\right]$$

$$\therefore \quad U = m \cdot G = \underbrace{m_1+m_3}_{U_1}, \underbrace{m_1+m_2}_{U_2}, \underbrace{m_2+m_3}_{U_3}, \underbrace{m_1}_{U_4}, \underbrace{m_2}_{U_5}, \underbrace{m_3}_{U_6}$$

Parity Check Matrix

For every $k \times n$ generator matrix there exists a parity check matrix H of dimension $(n-k) \times n$ such that the rows of G are orthogonal to the rows of H, i.e. $GH^T = 0$. This is given by

$$H = \left[I_{n-k} \vdots P^T \right]$$

$$\Rightarrow \quad H^T = \begin{bmatrix} I_{n-k} \\ \cdots \\ P \end{bmatrix}$$

or $H^T = \begin{bmatrix} 1 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 \\ P_{11} & P_{12} & \cdots & P_{1(n-k)} \\ P_{21} & P_{22} & \cdots & P_{2(n-k)} \\ \vdots & \vdots & & \vdots \\ P_{k1} & P_{k2} & \cdots & P_{k(n-k)} \end{bmatrix}$

This means that any codeword received, when multiplied with H^T must give the result as zero. This can be used as a technique to verify whether the code is received correctly or there is an error.

Proof:

$$U = m \cdot G.$$
$$\Rightarrow U H^T = m G \cdot H^T$$
$$= m \cdot 0$$
$$= 0$$

Thus U is a codeword generated by G if and only if $UH^T = 0$.

*Important Block Code: Hamming Code

It is a class of Linear Block Code characterized by

$$\langle n, k \rangle = \langle 2^m - 1, 2^m - 1 - m \rangle$$

where $m = 2, 3, 4 \ldots$

Distance between 2 block codes is written as Hamming Distance and is defined by the no. of uncommon bits between the codes. For eg. Distance between 110010 & 101100 would be 4.

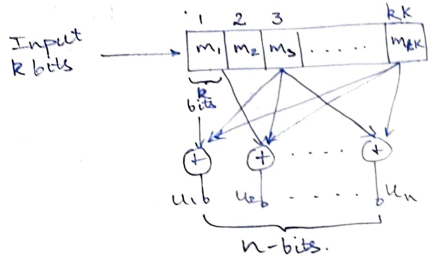
Hamming Weight is defined as the number of non-zero elements in it.

Convolution Codes

A convolution code is described by 3 integers, n, k and K. where the ratio k/n has the same code rate significance (information encoded per bit) that it has for the block codes BUT n does not define a block or codeword length as it does for block codes. It represents the no. of output bits for each k-bit message given to the system. The integer K is a parameter known as the constraint length which represents the number of k-tuple message stages in the encoding shift register. An important characteristic of convolutional codes, different from block codes, is that the encoder has memory — the n-tuple emitted by the convolutional encoding procedure is not only a function of an input k-tuple, but also a function of the previous K-1 input k-tuples. In practice, n and k are small integers and K is varied to control the capability and complexity of the code.

Assuming that each input message bit is equally likely and the message bits are represented by a vector 'm' the each sequence 'm' is transformed into a unique codeword sequence $U = G(m)$. The key feature here is that each output sequence U is not uniquely mapped or dependent on different k-tuple messages, but also on (K-1) input k tuples that precede it. This will be more clear with an example.

A typical convolution encoder looks as follows:



Let us take an example of a (2, 1) code with K = 3

⇒ n = 2 (2 mod 2 adders)

k = 1 (1 input bit enters the encoder at a time)

K = 3 (constraint length is 3)

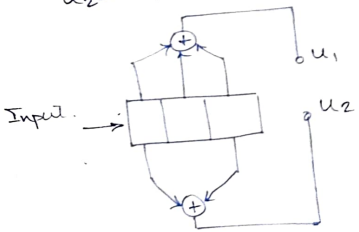
Also Let us say that $u_1 = m_1 \oplus m_2 \oplus m_3$

and $u_2 = m_1 \oplus m_3$

Then the encoder can be designed as follows.

1. Length of Shift register = 1 × 3 = 3
2. $u_1 = m_1 \oplus m_2 \oplus m_3$

   $u_2 = m_1 \oplus m_3$



One way to represent the encoder is to specify a set of n connection vectors, one for each of the mod-2 adders. Each vector has dimensions K and describes the connection of the encoding shift register to that modulo-2 adder. A one in the i th position of the vector indicates that the corresponding stage in the shift register is connected to the mod-2 adder and a zero in the given position indicates that no connection exists between the stage and the mod-2 adder. For the previous example, the connection vector g_1 for the code bit u_1 and the connection vector g_2 for the code bit u_2 can be written as (129)

$g_1 = 1\ 1\ 1$

$g_2 = 1\ 0\ 1$

A short note on RS Codes (Reed-Solomon Codes)

Before we understand RS codes let us define what is meant by a cyclic code. A (n, K) linear code is called a cyclic code if it can be described by the following property. If the n-tuple $U = (u_0, u_1, u_2 \ldots u_{n-1})$ is a codeword in the subspace S, then $U^{(1)} = (u_{n-1}, u_0, u_1, \ldots u_{n-2})$ obtained by an end around shift is also a codeword in S.

RS codes are NON BINARY cyclic codes with symbols made up of m-bit sequences, where m is a positive integer having a value greater than 2. R-S (n, K) codes exist for all m bit symbols where

$$0 < K < n < 2^m + 2$$

where K is the total number of data symbols being encoded and n is the total number of code symbols in the encoded

For the conventional R-S (n,k) code

$$(n,k) = (2^m - 1, 2^m - 1 - 2t)$$

where t is the symbol error correcting capability of the code and $n-k = 2t$ is the number of parity symbols.

Reed Solomon (R-S) codes acheive the largest possible code minimum distance for any linear code with the same encoder input and output block lengths. For nonbinary codes, the distance between two codewords is defined as the number of symbols in which the sequence differ. For R-S codes, the distance is given by

$$d_{min} = n - k + 1$$

The code is capable of correcting any combination of t or fewer errors, where t, obtained from a study of maximum likelihood error detection and correction capability, can be written as

$$t = \left\lfloor \frac{d_{min} - 1}{2} \right\rfloor = \left\lfloor \frac{n-k}{2} \right\rfloor$$

where $\lfloor x \rfloor$ represents largest integer less than or equal to x (floor function).

Intuitively it can be said that for each error, one symbol is required to detect the error a second one is required to correct it, i.e. for each error we must add 2 redundant symbols for accurate error detection.

For the case when we use $R = C$, we get

$$\frac{S}{N_o} = \left(\frac{E_b}{N_o}\right)\left(\frac{C}{W}\right)$$

$$\Rightarrow \frac{C}{W} = \log_2\left(1 + \frac{E_b}{N_o}\left(\frac{C}{W}\right)\right)$$

or $$2^{C/W} = 1 + \frac{E_b}{N_o}\left(\frac{C}{W}\right)$$

or $$\frac{E_b}{N_o} = \frac{W}{C}\left(2^{C/W} - 1\right)$$

Shannon's work showed that the values of S, N and W set a limit on the transmission bit rate but not on error probability.

Shannon Limit

There exist a limiting value of $\frac{E_b}{N_o}$ beyond which there can be no error free communication at any information rate.

Let us observe that

$$\frac{C}{W} = \log_2\left(1 + \frac{E_b}{N_o}\left(\frac{C}{W}\right)\right)$$

we know the mathematical identity

$$\lim_{x \to 0} (1+x)^{1/x} = e.$$

Thus we may rewrite the above equation as

$$\frac{C}{W} = \frac{E_b}{N_o}\left(\frac{C}{W}\right)\log_2\left[\left(1 + \frac{E_b}{N_o}\left(\frac{C}{W}\right)\right)^{1/\left(\frac{E_b}{N_o}\left(\frac{C}{W}\right)\right)}\right]$$

Shannon-Hartley Capacity Theorem

With the discussion on the channel encoding it becomes apparent to discuss what amount of bits can be tolerated over the channel of given bandwidth with a known SNR value. Shannon showed that the capacity C of a channel (theoretical maximum bit rate) perturbed by Additive White Gaussian Noise (AWGN) can be given by

$$C = W \log_2\left(1 + \frac{S}{N}\right)$$

where W = bandwidth
S = Avg. signal power
N = Avg. Noise power.

When W is in Hz and the log is taken to the base 2, the capacity is given in bits/sec. This means that the signal transmission can be done at any bit rate $R \leq C$ with a arbitrarily small error probability and not possible otherwise.

Relationship between $\frac{E_b}{N_0}$ and channel capacity can be formed as follows.

$$E_b = S \cdot T_b = S/R \quad \left(\begin{array}{l} T_b = \text{Time period of one bit} \\ R = \text{bit rate} \end{array}\right)$$

and

$$N_0 = N/W$$

$$\Rightarrow \frac{E_b}{N_0} = \left(\frac{S}{N}\right)\left(\frac{W}{R}\right)$$

or $$\frac{S}{N} = \frac{E_b}{N_0}\left(\frac{R}{W}\right)$$

This means

$$1 = \frac{E_b}{N_0} \log_2\left(1 + x\right)^{1/x}$$

where $x = \frac{E_b}{N_0}\left(\frac{C}{W}\right)$

∴ to find the limiting value of $\frac{E_b}{N_0}$, assuming that $x \to 0$ i.e. $\frac{E_b}{N_0}\left(\frac{C}{W}\right) \to 0$

$$1 = \frac{E_b}{N_0} \log_2\left(\lim_{x \to 0} (1+x)^{1/x}\right)$$

$$\Rightarrow \frac{E_b}{N_0} = \frac{1}{\log_2 e} = 0.693$$

In decibles, $\frac{E_b}{N_0} = -1.59\ dB \simeq -1.6\ dB$.

This value of E_b/N_0 is called the Shannon limit. It is not possible to reach the Shannon limit as this would mean that $\frac{C}{W} \to 0$ or $W \to \infty$ (as C can have any value) which in turn means that the message bits required for transmission $K \to \infty$.

Entropy

To design a communication system with a specified message handling capability, we need a metric for measuring the information content to be transmitted. Shannon developed such a metric, H, called the entropy of the message source.

Entropy is defined as the average amount of information per source output. For a message source with n possible outputs, entropy can be expressed as

$$H = -\sum_{i=1}^{n} P_i \log_2 P_i \text{ bits/source output}$$

where P_i is the probability of ith output and $\sum P_i = 1$

For example,
A binary message or a source having only two possible outputs with probability p and $q = (1-p)$, then the entropy is written as

$$H = -(p \log_2 p + q \log_2 q)$$

Example

(a) Calculate the average information in bits/character for the English language, assuming that each of the 26 characters in the alphabet occurs with equal likelihood.

$$H = -\sum_{i=1}^{26} \frac{1}{26} \log_2\left(\frac{1}{26}\right) = \log_2 26 = 4.7 \text{ bits/char}$$

(b) If $P = 0.10$ for a, e, o, t
$P = 0.07$ for h, i, n, s, r,
$P = 0.02$ for c, d, f, l, m, p, u, y
$P = 0.01$ for b, g, j, k, q, v, w, x, z

Calculate the entropy.

$$H = -(4 \times 0.1 \log_2 0.1 + 5 \times 0.07 \log 0.07 + 8 \times 0.02 \log 0.02 + 9 \times 0.01 \log 0.01)$$

$$= -4.17 \text{ bits/char}$$

Source Entropy

The idea behind source entropy is to find the average self information for the symbols in the alphabet. If we know the probability of transmission of each alphabet or symbol, then the source entropy would be given by

$$H(X) = -\sum_{j=1}^{N} P_j \log_2(P_j)$$

where N is the number of inputs and P_j is probability of the jth input. It can be shown that the range of $H(x)$ lies between 0 (in case of no uncertainity) and $\log_2(N)$ (in case of maximum uncertainity) i.e. all symbols or inputs are equiprobable)

$$\Rightarrow \quad 0 \leq H(x) \leq \log_2 N$$

Source entropy is also the measure of the minimum average code length achievable by variable length source encoding schemes.

Source Coding for Digital Data

Source coding is done to reduce the redundancies that are usually associated with the digital data. Generally any form of multi-symbol source codeword can be represented using a binary representation scheme. This introduces a lot of patterns and unwanted repetitions in the net transmission. We are going to discuss the principle of source coding for data compression and that will be done by appropriately reducing the number of bits. Data compression codes are often variable length codes.

Properties of Codes

① **Uniquely Decodable**

These codes which allow us to invert the mapping to the original symbol alphabet. Thus, each symbol is allotted a unique binary sequence.

② **Prefix free**

No codeword should be a prefix to any other codeword.

Consider the following example.

Symbol	Code 1	Code 2	Code 3	Code 4	Code 5	Code 6
a	00	00	0	1	1	1
b	00	01	1	10	00	01
c	11	10	11	100	01	11

Here code 1 is not uniquely decodable and codes 3, 4 and 6 are not prefix free. Thus only code 2 and code 5 are practically efficient.

Average Code length of Variable Length codes

It is defined as the ~~weighted~~ sum of the length of each codeword weighted by the probability of its occurance.

$$\Rightarrow \quad \bar{n} = \sum_{i=1}^{N} n_i P(x_i)$$

Huffman Code

Huffman Code is a prefix free variable length code that can achieve the ~~maximum~~ minimum avg. code length $\bar{n}$ for a given input alphabet.

* The minimum average codelength for a particular alphabet may be significantly greater than the source entropy. But this inability to exploit the maximum data compression is related to the alphabet and not to the source coding technique.

Compression performance is measured by the compression ratio. This ratio is equal to the average number of bits before compression to the average number of bits per sample after compression.

Procedure

① List the input alphabet symbols along with their probabilities in descending order of occurance.

② Two entries of the lowest probabilities are merged ~~to form a new branch with~~ their composite probability.

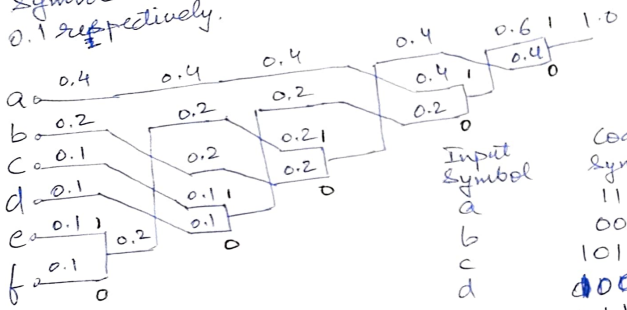
③ New branch and remaining branches are again reordered to assure that the reduced table preserves the descending probability. This is called "bubbling". In case the new branch has probability equal to existing branches, then the new branch is kept at the top of the group of all such branches.

④ Repeat ② and ③ until you arrive at a single branch with probability = 1.

⑤ Label all branches going up by a 1 and all branches going down by a 0. Trace the path back from right to left and the binary sequence that corresponds to each branch is its codeword

Example-1

Symbols a, b, c, d, e, f with probabilities 0.4, 0.2, 0.1, 0.1, 0.1, 0.1 respectively.



Input Symbol	Code Symbol
a	11
b	00
c	101
d	100
e	011
f	010

Finding the average code length

X_i	$P(X_i)$	Code	n_i	$n_i P(X_i)$
a	0.4	11	2	0.8
b	0.2	00	2	0.4
c	0.1	101	3	0.3
d	0.1	100	3	0.3
e	0.1	011	3	0.3
f	0.1	010	3	0.3
			$\bar{n}$ =	2.4

⇒ Compression ratio = 3/2.4 = 1.25

(3 because 6 symbols require 3 bits atleast)

Also, Source entropy of the system = 2.32.

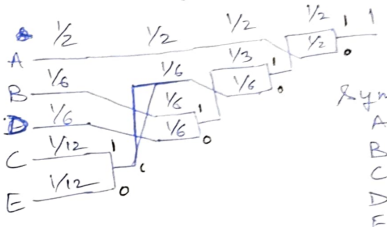
⇒ (2.32/2.40) = 96.7% of the possible compression ratio has been achieved.

* Avg. length of 2.40 indicates that for 100 symbols, 240 bits need to be sent instead of 300.

Example 2

Find Huffman's code for 5 symbols with probability of $P(A) = \frac{1}{2}$, $P(B) = \frac{1}{6}$, $P(C) = \frac{1}{12}$, $P(D) = \frac{1}{6}$ and $P(E) = \frac{1}{12}$.

Calculate Coding Efficiency.



Symbol	Codeword
A	0
B	111
C	101
D	110
E	100

Average Code length

Symbol	Probability $P(X_i)$	Codeword	Code length (n_i)	$n_i X_i$
A	1/2	0	1	0.5
B	1/6	111	3	0.5
C	1/12	101	3	0.25
D	1/6	110	3	0.5
E	1/12	100	3	0.25
			$\bar{n}$ =	2

Source Entropy = 1.96 bits/symbol.

⇒ Compression ratio = 3/2 = 1.5 which is (1.96/2) = 98% efficient.