UNIT IV INFORMATION THEORY AND CODING

ENTROPY, SOURCE ENCODING THEOREM

Source encoding theorem

The discrete memory less source of entropy H(X), the average code word length (L) for any distortion less source encoding is bounded.

Code redundancy is the measure of redundancy bits in the encoded message sequence.

Mutual information is the amount of information transferred when Xi is transmitted and Yi is received. It is represented by I(Xi,Yi) .The average mutual information is defined as the amount of source information gain per received symbol.

A block code of length n and 2k code words is calleda linear (n, k) code if and only if its 2k code words form a k-dimensional subspace of the vector space of all the n-tuples over the field GF(2). The message occurring frequently can be assigned short code words, whereas message which occur rarely are assigned long code word, such coding is called variable length coding.

The efficient representation of data generated by a discrete source is known as source encoding. This device that performs this representation is called source encoder.

The types of error control method

Error detection and retransmission

Error detection and correction

Channel capacity is defined as the maximum of the mutual information that may be transmitted through the channel.

The needs for encoding

To improve the efficiency of communication

To improve the transmission quality.

The entropy of a source is a measure of the average amount of information per source symbol in a long message. Channel coding theorem is applied for discrete memory less additive white gaussian noise channels.

The advantages of Shannon fano coding

Reduced bandwidth

Reduced noise

It can be used for error detection and correction.

The objectives of cyclic codes :

Encoding and syndrome calculations can be easily implemented by using simple shift register with feedback connection It is possible to design codes having useful error correction properties

Source Coding Theorem

The **source coding theorem** shows that (in the limit, as the length of a stream of independent and identically-distributed random variable (data tends to infinity) it is impossible to compress the data such that the code rate (average number of bits per symbol) is less than the Shannon entropy of the source, without it being virtually certain that information will be lost. However it is possible to get the code rate arbitrarily close to the Shannon entropy, with negligible probability of loss.

Proof: Source coding theorem

Given X is an i.i.d. source, its time series X_1, ..., X_n is i.i.d. with entropy $H(X)$ in the discrete-valued case and differential entropy in the continuous-valued case. The Source coding theorem states that for any $\varepsilon > 0$ for any rate larger than the entropy of the source, there is large enough n and an encoder that takes n i.i.d. repetition of the source, $X^{1:n}$, and maps it to $n(H(X) + \varepsilon)$ binary bits such that the source symbols $X^{1:n}$ are recoverable from the binary bits with probability at least $1 - \varepsilon$.

Proof of Achievability. Fix some $\varepsilon > 0$, and let

$$p(x_1, \ldots, x_n) = \Pr\left[X_1 = x_1, \cdots, X_n = x_n\right].$$

The typical set, *Aεn*, is defined as follows:

$$A_n^\varepsilon = \left\{ (x_1, \cdots, x_n) \ : \ \left| -\frac{1}{n} \log p(x_1, \cdots, x_n) - H_n(X) \right| < \varepsilon \right\}.$$

The Asymptotic Equipartition Property (AEP) shows that for large enough *n*, the probability that a sequence generated by the source lies in the typical set, *Aεn*, as defined approaches one. In particular there for large enough *n*, $P(A_n^\varepsilon) > 1 - \varepsilon$(See AEP for a proof):

The definition of typical sets implies that those sequences that lie in the typical set satisfy:

$$2^{-n(H(X)+\varepsilon)} \leq p\left(x_1, \cdots, x_n\right) \leq 2^{-n(H(X)-\varepsilon)}$$

Note that:

- The probability of a sequence from *X* being drawn from *Aε n* is greater than 1 − *ε*.
- $|A_n^\varepsilon| \leq 2^{n(H(X)+\varepsilon)}$since the probability of the whole set *Aε n* is at most one.
- $|A_n^\varepsilon| \geq (1-\varepsilon)2^{n(H(X)-\varepsilon)}$. For the proof, use the upper bound on the probability of each term in typical set and the lower bound on the probability of the whole set *Aε n*.

Since $|A_n^\varepsilon| \leq 2^{n(H(X)+\varepsilon)}, n.(H(X)+\varepsilon)$bits are enough to point to any string in this set.

The encoding algorithm: The encoder checks if the input sequence lies within the typical set; if yes, it outputs the index of the input sequence within the typical set; if not, the encoder outputs an arbitrary *n*(*H*(*X*) + *ε*) digit number. As long as the input sequence lies within the typical set (with probability at least 1 − *ε*), the encoder doesn't make any error. So, the probability of error of the encoder is bounded above by *ε*.

Proof of Converse. The converse is proved by showing that any set of size smaller than *Aε*
n (in the sense of exponent) would cover a set of probability bounded away from 1.

Proof: Source coding theorem for symbol codes

For $1 \le i \le n$ let s_i denote the word length of each possible x_i. Define $q_i = a^{-s_i}/C$, where C is chosen so that $q_1 + ... + q_n = 1$. Then

$$\begin{aligned} H(X) &= -\sum_{i=1}^{n} p_i \log_2 p_i \\ &\le -\sum_{i=1}^{n} p_i \log_2 q_i \\ &= -\sum_{i=1}^{n} p_i \log_2 a^{-s_i} + \sum_{i=1}^{n} p_i \log_2 C \\ &= -\sum_{i=1}^{n} p_i \log_2 a^{-s_i} + \log_2 C \\ &\le -\sum_{i=1}^{n} -s_i p_i \log_2 a \\ &\le \mathbb{E} S \log_2 a \end{aligned}$$

where the second line follows from Gibbs' inequality and the fifth line follows from Kraft's inequality:

$$C = \sum_{i=1}^{n} a^{-s_i} \le 1$$

so $\log C \le 0$.

For the second inequality we may set

$$s_i = \lceil -\log_a p_i \rceil$$

so that

$$-\log_a p_i \le s_i < -\log_a p_i + 1$$

and so

$$a^{-s_i} \le p_i$$

and

$$\sum a^{-s_i} \le \sum p_i = 1$$

and so by Kraft's inequality there exists a prefix-free code having those word lengths. Thus the minimal S satisfies

$$\begin{aligned} \mathbb{E} S &= \sum p_i s_i \\ &< \sum p_i \left(-\log_a p_i + 1\right) \\ &= \sum -p_i \frac{\log_2 p_i}{\log_2 a} + 1 \\ &= \frac{H(X)}{\log_2 a} + 1 \end{aligned}$$

Techniques used for compression of information.

Shannon Fano Coding Techniques

In the field of data compression, **Shannon-Fano coding** is a suboptimal technique for constructing a prefix code based on a set of symbols and their probabilities (estimated or measured).

In Shannon-Fano coding, the symbols are arranged in order from most probable to least probable, and then divided into two sets whose total probabilities are as close as possible to being equal.All symbols then have the first digits of their codes assigned; symbols in the first set receive "0" and symbols in the second set receive "1". As long as any sets with more than one member remain, the same process is repeated on those sets, to determine successive digits of their codes. When a set has been reduced to one symbol, of course, this means the symbol's code is complete and will not form the prefix of any other symbol's code.

The algorithm works, and it produces fairly efficient variable-length encodings; when the two smaller sets produced by a partitioning are in fact of equal probability, the one bit of information used to distinguish them is used most efficiently. Unfortunately, Shannon-Fano does not always produce optimal prefix codes; the set of probabilities {0.35, 0.17, 0.17, 0.16, 0.15} is an example of one that will be assigned

Shannon-Fano Algorithm

A Shannon-Fano tree is built according to a specification designed to define an effective code table. The actual algorithm is simple:

1. For a given list of symbols, develop a corresponding list of probabilities or frequency counts so that each symbol‘s relative frequency of occurrence is known.
2. Sort the lists of symbols according to frequency, with the most frequently occurring symbols at the left and the least common at the right.

1. Divide the list into two parts, with the total frequency counts of the left half being as close to the total of the right as possible.

2. The left half of the list is assigned the binary digit 0, and the right half is assigned the digit 1. This means that the codes for the symbols in the first half will all start with 0, and the codes in the second half will all start with 1.
3. Recursively apply the steps 3 and 4 to each of the two halves, subdividing groups and adding bits to the codes until each symbol has become a corresponding code leaf on the tree.

Huffmann Coding Techniques

Huffman coding is an entropy encoding algorithm used for lossless data compression. The term refers to the use of a variable-length code table for encoding a source symbol (such as a character in a file)

Huffman coding uses a specific method for choosing the representation for each symbol, resulting in a prefix code (sometimes called "prefix-free codes") (that is, the bit string representing some particular symbol is never a prefix of the bit string representing any other symbol) that expresses the most common characters using shorter strings of bits than are used for less common source symbols. Huffman was able to design the most efficient compression method *of this type*: no other mapping of individual source symbols to unique strings of bits will produce a smaller average output size when the actual symbol frequencies agree with those used to create the code.

Although Huffman coding is optimal for a symbol-by-symbol coding (i.e. a stream of unrelated symbols) with a known input probability distribution, its optimality can sometimes accidentally be over-stated. For example, arithmetic coding and LZW coding often have better compression capability.

Given

A set of symbols and their weights (usually proportional to probabilities).

Find

A prefix-free binary code (a set of codewords) with minimum expected codeword length (equivalently, a tree with minimum weighted path length).

Input.

Alphabet , which is the symbol alphabet of size *n*.

Set , which is the set of the (positive) symbol weights (usually proportional to probabilities), i.e. .

Output.

Code , which is the set of (binary) codewords, where c_i is the codeword for .

Goal.

Let be the weighted path length of code *C*. Condition: for any code .

For any code that is *biunique*, meaning that the code is ***uniquely decodable***, the sum of the probability budgets across all symbols is always less than or equal to one. In this example, the sum is strictly equal to one; as a result, the code is termed a *complete* code. If this is not the case, you can always derive an equivalent code by adding extra symbols (with associated null probabilities), to make the code complete while keeping it *biunique*.

In general, a Huffman code need not be unique, but it is always one of the codes minimizing $L(C)$.

MUTUAL INFORMATION

On an average we require $\boldsymbol{H(X)}$ bits of information to specify one input symbol. However, if we are allowed to observe the output symbol produced by that input, we require, then, only $\boldsymbol{H(X|Y)}$ bits of information to specify the input symbol. Accordingly, we come to the conclusion, that on an average, observation of a single output provides with $[\boldsymbol{H(X) - H(X|Y)}]$ bits of information. This difference is called ‗ ***Mutual Information***‘ or ‗ ***Transinformation***‘ of the channel, denoted by $\boldsymbol{I(X, Y)}$. Thus:

$$I(X, Y) = H(X) - H(X|Y)$$

Notice that in spite of the variations in the source probabilities, $\boldsymbol{p(x_k)}$ (may be due to noise in the channel), certain probabilistic information regarding the state of the input is available, once the conditional probability $\boldsymbol{p(x_k \mid y_j)}$ is computed at the receiver end. The difference between the initial uncertainty of the source symbol x_k, i.e. $\boldsymbol{\log 1/p(x_k)}$ and the final uncertainty about the same source symbol x_k, after receiving y_j, **i.e.** $\boldsymbol{\log 1/p(x_k|y_j)}$ is the information gained through the channel. This difference we call as the mutual information.

mutual information is always non- negative‖. That is to say, we cannot loose information on an average by observing the output of a channel. An easy method, of remembering the various relationships, is given in Fig 4.2.Althogh the diagram resembles a Venn-diagram, it is not, and the diagram is only a tool to remember the relationships. That is all. You cannot use this diagram for proving any result.

The entropy of ***X*** is represented by the circle on the left and that of ***Y*** by the circle on the right. The overlap between the two circles (dark gray) is the mutual information so that the remaining (light gray) portions of ***H*(*X*)** and ***H*(*Y*)** represent respective equivocations. Thus we have

$$H(X \mid Y) = H(X) - I(X, Y) \text{ and } H(Y \mid X) = H(Y) - I(X, Y)$$

The joint entropy ***H*(*X,Y*)** is the sum of ***H*(*X)*** and ***H*(*Y*)** except for the fact that the overlap is added twice so that

$$H(X, Y) = H(X) + H(Y) - I(X, Y)$$

Also observe $H(X, Y) = H(X) + H(Y|X)$

$$= H(Y) + H(X|Y)$$

For the **JPM** given by I(***X, Y*)** **= 0.760751505 bits / sym**

ENTROPY:

The different conditional entropies, Joint and Conditional Entropies:

It is clear that all the probabilities encountered in a two dimensional communication system could be derived from the **JPM**. While we can compare the **JPM**, therefore, to the impedance or admittance matrices of an ***n***-port electric network in giving a unique description of the system under consideration, notice that the **JPM** in general, need not necessarily be a square matrix and even if it is so, it need not be symmetric.

We define the following entropies, which can be directly computed from the **JPM**.

$$H(X,Y) = p(x_1, y_1)\log\frac{1}{p(x_1,y_1)} + p(x_1,y_2)\log\frac{1}{p(x_1,y_2)} + \ldots + p(x_1,y_n)\log\frac{1}{p(x_1,y_n)}$$

$$+\, p(x_2,y_1)\log\frac{1}{p(x_2,y_1)} + p(x_2,y_2)\log\frac{1}{p(x_2,y_2)} + \ldots + \log\; p(x_2,y_n)\frac{1}{p(x_1,y_1)}$$

$$+ \ldots\; p(x_m,y_1)\log\frac{1}{p(x_m,y_1)} + p(x_m,y_2)\log\frac{1}{p(x_m,y_2)} + \ldots\; p(x_m,y_n)\log\frac{1}{p(x_m,y_n)}$$ or

Observe that the manipulations, ‗ The entropy you want is simply the double summation of joint probability multiplied by logarithm of the reciprocal of the probability of interest‟ .

For example, if you want joint entropy, then the probability of interest will be joint probability.

If you want source entropy, probability of interest will be the source probability. If you

want the equivocation or conditional entropy, H (X|Y) then probability of interest will be the conditional probability p (xK |yj) and so on. All the five entropies so defined are all inter-related.

H(Y | X) = H(X, Y) – H(X)

H(X, Y) = H(X) + H(Y | X)

H(X, Y) = H(Y) + H(X | Y)

Or $$H(Y \mid X) = H(X, Y) - H(X)$$

That is: $$H(X, Y) = H(X) + H(Y \mid X)$$

Similarly, you can show: $H(X, Y) = H(Y) + H(X \mid Y)$

Consider $H(X) - H(X \mid Y)$. We have:

$$H(X) - H(X|Y) = \sum_k \sum_j p(x_k, y_j) \log \frac{1}{p(x_k)} \log \frac{1}{p(x_k \mid y_j)}$$

$$= \sum_k \sum_j p(x_k, y_j) \log \frac{p(x_k, y_j)}{p(x_k) \cdot p(y_j)}$$

Using the logarithm inequality derived earlier, you can write the above equation as:

$$H(X) - H(X|Y) = \log e \sum_k \sum_j p(x_k, y_j) \ln \frac{p(x_k, y_j)}{p(x_k) \cdot p(y_j)}$$

$$\geq \log e \sum_k \sum_j p(x_k, y_j)\ 1 - \frac{p(x_k) \cdot p(y_j)}{p(x_k, y_j)}$$

$$\geq \log e \sum_k \sum_j p(x_k, y_j) - \sum_{kj}\sum p(xk).p(yj)$$

$$\geq \log e \sum_k \sum_j p(x_k, y_j) - \sum_k p(xk).\sum_j p(yj) \geq 0$$

Because $\sum_k \sum_j p(x_k, y_j) \quad \sum_k p(x_k) \quad \sum_j p(y_j) = 1$. Thus it follows that:

$$H(X) \geq H(X|Y)$$

Similarly, $H(Y) - H(Y|X)$

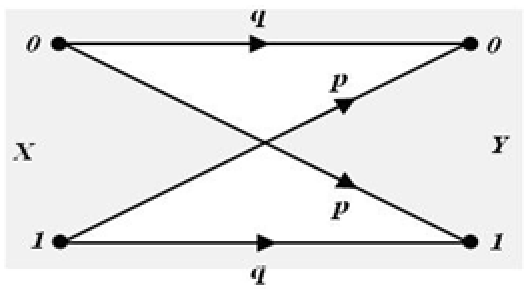
Equality in holds iffy P $(x_k, y_j) = p(x_k) .p(y_j)$; *i*.e., **if and only if input symbols and output symbols are statistically independent of each other.**

Binary Symmetric Channels (BSC):

The channel is called a ‗Binary Symmetric Channel' or (***BSC***). It is one of the most common and widely used channels. The channel diagram of a ***BSC*** is shown in Fig 3.4. Here ‗ ***p***' is called the error probability.

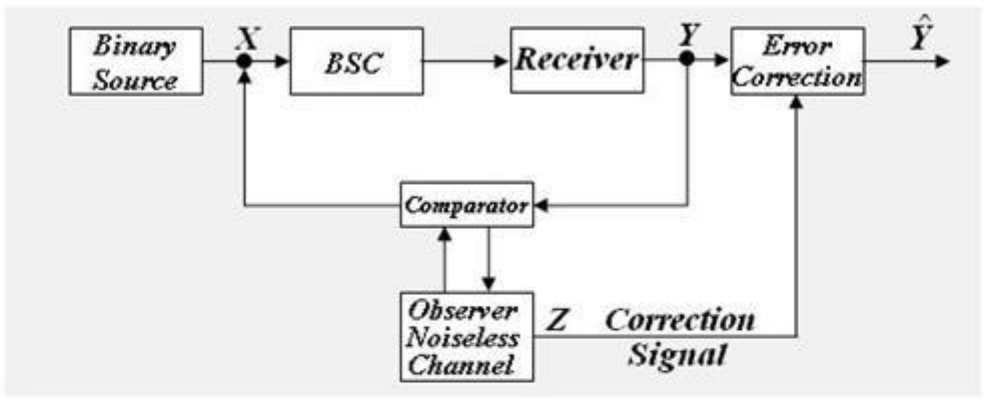
For this channel we have:

In this case it is interesting to note that the equivocation, ***H* (*X*|*Y*) =*H* (*Y*|*X*).**



$$P(Y|X) = \begin{array}{c c} & \begin{array}{cc} 0 & 1 \end{array} \\ \begin{array}{c} 0 \\ 1 \end{array} & \begin{bmatrix} q & p \\ q & p \end{bmatrix} \end{array} \qquad \begin{array}{l} p + q = 1 \\ P(X{=}0) = \alpha \\ P(X{=}1) = 1 - \alpha \end{array}$$

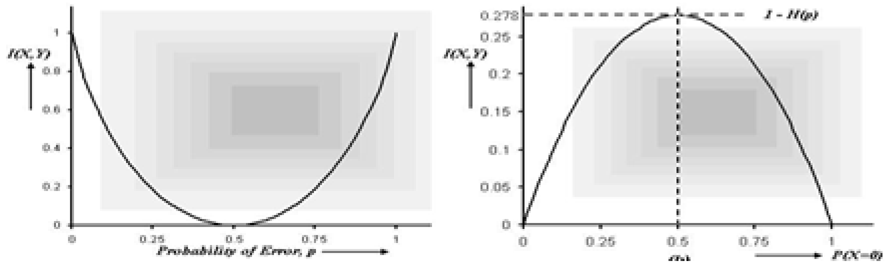
An interesting interpretation of the equivocation may be given if consider an idealized communication system with the above symmetric channel.



The observer is a noiseless channel that compares the transmitted and the received symbols. Whenever there is an error a ‗ 1' is sent to the receiver as a correction signal and appropriate correction is effected. When there is no error the observer transmits a ‗ 0' indicating no change. Thus the observer supplies additional information to the receiver, thus compensating for the noise in the channel. Let us compute this additional information .With P (X=0) = P (X=1) = 0.5, we have:

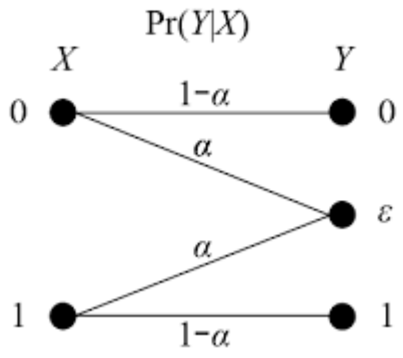
Probability of sending a „1" = Probability of error in the channel .
Probability of error = P (Y=1|X=0).P(X=0) + P (Y=0|X=1).P(X=1) = p × 0.5 + p × 0.5 = p

Thus the additional information supplied by the observer is exactly equal to the equivocation of the source. Observe that if ‗ ***p***' and ‗ ***q***' are interchanged in the channel matrix, the trans - information of the channel remains unaltered. The variation of the mutual information with the probability of error is
shown in Fig 3.6(*a*) for ***P (X=0) = P (X=1) = 0.5***. In Fig 4.6(*b*) is shown the dependence of the mutual information on the source probabilities.

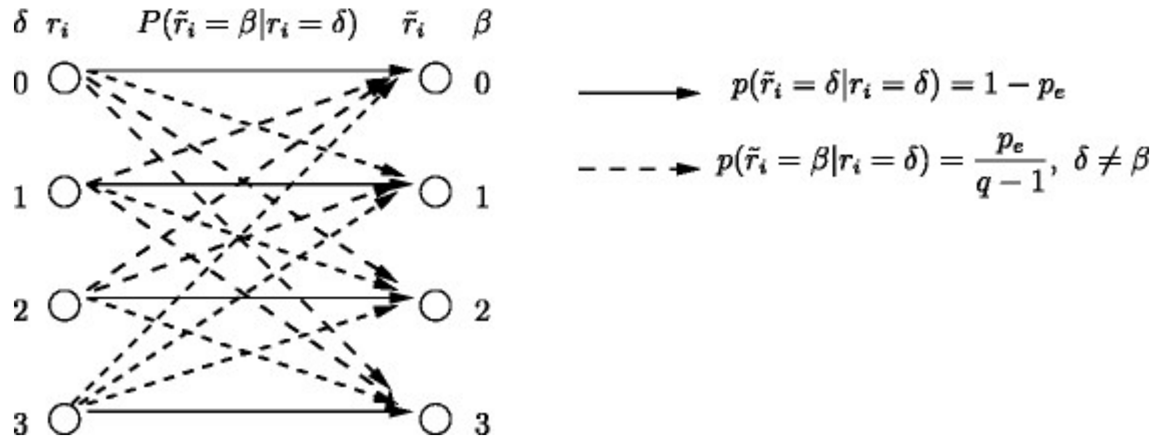


Binary Erasure Channels (*BEC*):

BEC is one of the important types of channels used in digital communications. Observe that whenever an error occurs, the symbol will be received as ‗ ***y***' and no decision will be made about the information but an immediate request will be made for retransmission, rejecting what have been received (***ARQ*** techniques), thus ensuring ***100%*** correct data recovery.



$$\Pr[Y=0|X=0]=1-P_e$$
$$\Pr[Y=0|X=1]=0$$
$$\Pr[Y=1|X=0]=0$$
$$\Pr[Y=1|X=1]=1-P_e$$
$$\Pr[Y=e|X=0]=P_e$$
$$\Pr[Y=e|X=1]=P_e$$


δ r_i
$P(\tilde{r}_i=\beta|r_i=\delta)$
$\tilde{r}_i$ β
0
1
2
3
0
1
2
3
$p(\tilde{r}_i=\delta|r_i=\delta)=1-p_e$
$p(\tilde{r}_i=\beta|r_i=\delta)=\frac{p_e}{q-1},\ \delta\neq\beta$

C = Max I (X, Y) = q bits / symbol.

In this particular case, use of the equation ***I(X, Y) = H(Y) – H(Y | X)*** will not be correct, as ***H(Y)*** involves ‗***y***‘ and the information given by ‗***y***‘ is rejected at the receiver.

4.6 Channel Capacity theorem
Shannon's theorem: on channel capacity("coding Theo rem")

It is possible, in principle, to device a means where by a communication system will transmit information with an arbitrary small probability of error, provided that the information rate ***R***(=***r***×***I*** (***X***,***Y***),where ***r*** is the symbol rate) is less than or equal to a rate ‗ ***C***‘ called ―channel capacity‖.

The technique used to achieve this objective is called coding. To put the matter more formally, the theorem is split into two parts and we have the following statements.

Positive statement:

*– Given a source of **M** equally likely messages, with **M>>1**, which is generating information at a rate **R**, and a channel with a capacity **C.** If **R ≤ C**, then there exists a coding technique such that the output of the source may be transmitted with a probability of error of receiving the message that can be made arbitrarily small‖.*

This theorem indicates that for ***R*≤ *C*** transmission may be accomplished without error even in the presence of noise. The situation is analogous to an electric circuit that comprises of only pure capacitors and pure inductors. In such a circuit there is no loss of energy at all as the reactors have the property of storing energy rather than dissipating.

Negative statement:

– *Given the source of **M** equally likely messages with **M>>1**, which is generating information at a rate **R** and a channel with capacity **C**. Then, if **R>C**, then the probability of error of receiving the message is close to unity for every set of **M** transmitted symbols‖.*

This theorem shows that if the information rate ***R*** exceeds a specified value ***C***, the error probability will increase towards unity as ***M*** increases. Also, in general, increase in the complexity of the coding results in an increase in the probability of error. Notice that the situation is analogous to an electric network that is made up of pure resistors. In such a circuit, whatever energy is supplied, it will be dissipated in the form of heat and thus is a –lossy network‖.

You can interpret in this way: Information is poured in to your communication channel. You should receive this without any loss. Situation is similar to pouring water into a tumbler. Once the tumbler is full, further pouring results in an over flow. You cannot pour water more than your tumbler can hold. Over flow is the loss.

Shannon defines –C‖ the channel capacity of a communication channel a s the maximum value of Transinformation, ***I*(*X, Y*):**

$$C \overset{\Delta}{=} \mathrm{Max}\ I(X, Y) = \mathrm{Max}\ [H(X) - H(Y|X)]$$

The maximization in Eq (4.28) is with respect to all possible sets of probabilities that could be assigned to the input symbols. Recall the maximum power transfer theorem: ‗In any network, maximum power will be delivered to the load only when the load and the source are properly matched‘. The device used for this matching purpose, we shall call a –transducer –. For example, in a radio receiver, for optimum response, the impedance of the loud speaker will be matched to the impedance of the output power amplifier, through an output transformer.

This theorem is also known as –The Channel Coding Theorem‖ (Noisy Coding Theorem). It may be stated in a different form as below:

$$R \leq C \text{ or } r_s H(S) \leq r_c I(X,Y)_{Max} \text{ or}\{ H(S)/T_s\} \leq \{ I(X,Y)_{Max}/T_c\}$$

“If a discrete memoryless source with an alphabet ‘S’ has an entropy H(S) and produces symbols every ‘T $_s$’ seconds; and a discrete memoryless channel has a capacity $I(X,Y)_{Max}$ and is used once every T_c seconds; then if

$$\frac{H(S)}{T_s} \quad \frac{I(X,Y)_{Max}}{T_c}$$

There exists a coding scheme for which the source output can be transmitted over the channel and be reconstructed with an arbitrarily small probability of error. The parameter C/T_c is called the critical rate. When this condition is satisfied with the equality sign, the system is said to be signaling at the critical rate.

channel and reconstruct it with an arbitrarily small probability of error

A communication channel, is more frequently, described by specifying the source probabilities ***P(X)*** & the conditional probabilities ***P (Y|X)*** rather than specifying the **JPM**. The **CPM**, ***P (Y|X),*** is usually referred to as the ‗ ***noise characteristic***‘ of the channel. Therefore unless otherwise specified, we shall understand that the description of the channel, by a matrix or by a ‗Channel diagram‘ always refers to **CPM, *P (Y|X)***. Thus, in a discrete communication channel with pre-specified noise characteristics (i.e. with a given transition probability matrix, ***P (Y|X))*** the rate of information transmission depends on the source that drives the channel. Then, the maximum rate corresponds to a proper matching of the source and the channel. This ideal characterization of the source depends in turn on the transition probability characteristics of the given channel.

Bandwidth-Efficiency: Shannon Limit:

In practical channels, the noise power spectral density N_0 is generally constant. If E_b is the transmitted energy per bit, then we may express the average transmitted power as:

$$S = E_b C$$

(C/B) is the –bandwidth efficiency‖ of the syste m. If $C/B = 1$, then it follows that $E_b = N_0$.This implies that the signal power equals the noise power. Suppose, $B = B_0$ for which, $S = N$, then Eq. (5.59) can be modified as:

That is, – *the maximum signaling rate for a given* ***S*** *is 1.443 bits/sec/Hz in the bandwidth over which the signal power can be spread without its falling below the noise level‖*.

4.7 CYCLIC CODES:

In coding theory, cyclic codes are linear block error-correcting codes that have convenient algebraic structures for efficient error detection and correction.

Let C be a linear code over a finite field $GF(q)^n$ of block length n. C is called a cyclic code, if for every codeword $c=(c_1,...,c_n)$ from C, the word $(c_n,c_1,...,c_{n-1})$ in $GF(q)^n$ obtained by a cyclic right shift of components is again a codeword. Same goes for left shifts. One right shift is equal to $n-1$ left shifts and vice versa. Therefore the linear code C is cyclic precisely when it is invariant under all cyclic shifts.

Cyclic Codes have some additional structural constraint on the codes. They are based on Galois fields and because of their structural properties they are very useful for error controls. Their structure is strongly related to Galois fields because of which the encoding and decoding algorithms for cyclic codes are computationally efficient.

Cyclic code for correcting error:

a) For correcting single error

The cyclic codes explicitly with error detection and correction. Cyclic codes can be used to correct errors, like Hamming codes as a cyclic codes can be used for correcting single error. Likewise, they are also used to correct double errors and burst errors. All types of error corrections are covered briefly in the further subsections.

The Hamming code has a generator polynomial $g(x)=x^3+x+1$. This polynomial has a zero in Galois extension field GF(8) at the primitive element α, and all codewords satisfy .

$C(\alpha)=0$ Cyclic codes can also be used to correct double errors over the field GF(2). Blocklength will be n equal to $2^m - 1$and primitive elements α and α^3as zeros in the $GF(2^m)$ because we are considering the case of two errors here, so each will represent one error.The received word is a polynomial of degree n-1 given as

$$v(x) = a(x)g(x) + e(x)$$

where e(x) can have at most two nonzero coefficients corresponding to 2 errors.

Syndrome Polynomial, S(x) as the remainder of polynomial v(x) when divided by the generator polynomial g(x) i.e.

S(x)=v(x) mod g(x)= (a(x)g(x)+e(x)) mod g(x)=e(x) mod g(x) as (a(x)g(x)) mod g(x) is zero

b) For correcting two errors

Let the field elements $X_1$and X_2 be the two error location numbers. If only one error occurs then $X_2$is equal to zero and if none occurs both are zero.

Let $S_1 = v(\alpha)$ and $S_3 = v(\alpha^3)$.

These field elements are called "syndromes". Now because $g(x)$is zero at primitive elements $\alpha$and α^3, so we can write $S_1 = e(\alpha)$ and $S_3 = e(\alpha^3)$. If say two errors occur, then

$$S_1 = \alpha^i + \alpha^{i'} \quad \text{and} \quad S_3 = \alpha^{3i} + \alpha^{3i'}.$$

And these two can be considered as two pair of equations in $GF(2^m)$with two unknowns and hence we can write

$S_1 = X_1 + X_2$ and $S_3 = (X_1)^3 + (X_2)^3$.

Hence if the two pair of nonlinear equations can be solved cyclic codes can used to correct two errors.

Syndrome Calculator

We know $r\,(X) = q(X)g(X) + S(X)$.

We can also write $r\,(X) = c(X) + e(X)$.

Rewrite the expression to:

$(X) = c(X) + r\,(X)$

$= c(X) + q(X)g(X) + S(X)$

$= (f\,(X) + q(X))g(X) + S(X)$

If error polynomial $e(X)$ is divided by generator polynomial $g(X)$, the remainder is the syndrome polynomial $S(X)$.

Error Detection

r (X) = c(X) + e(X) = q(X)g(X) + S(X)

e(X) = c(X) + q(X)g(X) + S(X) = (f (X) + q(X))g(X) + S(X)

Investigate error detecting capability of cyclic code:

Assuming $e(X)$ is a burst of length n - k or less, i.e., errors are confined to n - k or fewer consecutive positions;

$e(X)$ can be expressed by $e(X) = XjB(X)$, here $B(X)$ is a polynomial of degree $n - k - 1$ or less;

Xj cannot divided by $g(X)$, $B(X)$ cannot divided by $g(X)$ neither,

$e(X) = X^{jj}B(X)$ is NOT divisible by $g(X)$;

Thus the syndrome polynomial is not equal to zero.

It means that a cyclic code *Ccyc* (*n*,*k*) can detect any error burst of length *n* - *k* or less.

A cyclic code *Ccyc* (*n; k*) can also detect all the *end-around* error bursts of length *n* - *k* or less.

e = (1 1 0 0 0 0 0 0 1 0 1)

Cyclic Redundancy Check (CRC) codes:

Cyclic redundancy .check codes are extremely well suited for "error detection". The two important reasons for this statement are,

(1) they can be designed to detect many combinations of likely errors.

(2) The implementation of both encoding and error detecting circuits is practical.

Accordingly, all error detecting codes used in practice, virtually, are of theCRC -type. In ann-bit received word if a contiguous sequence of ‗b-bits‘ in which the first and the last bits and any number of intermediate bits are received in error, then we say aCRC "error burst' of length ‗b‘

has occurred. Such an error burst may also include an end-shifted version of the contiguous sequence.

In any event, Binary (n, k)CRC codes are capable of detecting the following error patterns:

1. All CRC error bursts of length (n-k) or less.
2. A fraction of (1 - 2 (n –k - 1)) of CRC error bursts of length (n – k + 1).
3. A fraction (1-2(n – k)) of CRC error bursts of length greater than (n – k + 1).
4. All combinations of (d min – 1) or fewer errors.
5. All error patterns with an odd number of errors if the generator polynomial

g (X) has an even number of non zero coefficients.

Generator polynomials of three CRC codes, internationally accepted as standards are listed below.

All three contain (1 +X) as a prime factor. TheCRC-12 code is used when the character lengths is 6- bits. The others are used for 8-bit characters.

* CRC-12 code: $g\ (X) = 1 + X + X^2+X^3 + X^{11} + X^{12}$*

*CRC-16 code: $g\ (X) = 1 +X^2 + X^{15} + X^{16}$

*CRC-CCITT code: $g\ (X) = 1 + X^5 + x^{12}+ X^{26}$.

Definition 1 An (n, k) linear block code C is said to be **cyclic** if for every code word $\mathbf{c} = (c_0, c_1, \ldots, c_{n-1})$ in C, there is also a code word $\mathbf{c}' = (c_{n-1}, c_0, \ldots, c_{n-2})$ that is also in C. ($\mathbf{c}'$ is a cyclic shift of $\mathbf{c}$.) □

It will be convenient to represent our codewords as *polynomials*. The codeword

$$\mathbf{c} = (c_0, c_1, \ldots, c_{n-1})$$

is represented by the polynomial

$$c(x) = c_0 + c_1 x + \cdots + c_{n-1} x^{n-1}$$

using the obvious one-to-one correspondence. A cyclic shift can therefore be represented as follows. Observe that

$$xc(x) = c_0 x + c_1 x + \cdots + c_{n-1} x^n.$$

If we not take this product modulo $x^n - 1$ we get

$$xc(x) \pmod{x^n - 1} = c_{n-1} + c_0 x + \cdots + x_{n-2} x^{n-1}.$$

So multiplication by x in the ring $GF(q)[x]/(x^n - 1)$ corresponds to a cyclic shift. Furthermore, any power of x times a codeword yields a codeword (apply the definition recursively), so that, for example,

$$\begin{gathered}(c_{n-1}, c_0, c_1, \ldots, c_{n-2}) \leftrightarrow xc(x) \\ (c_{n-2}, c_{n-1}, c_0, \ldots, c_{n-3}) \leftrightarrow x^2 c(x) \\ \vdots \\ (c_1, c_2, \ldots, c_{n-1}, c_0) \leftrightarrow x^{n-1} c(x)\end{gathered}$$

where the arithmetic is done in the ring $GF(q)[x]/(x^n - 1)$. Now observe that if we take an polynomial $a(x) \in GF(q)[x]$ of the form

$$a(x) = a_0 + a_1 x + \cdots + a_{n-1} x^{n-1}$$

then

$$c(x)a(a)$$

is simply a linear combination of cyclic shifts of $c(x)$ and hence, must also be a codeword. Hence: **a cyclic code is an ideal** in $GF(q)[x]/(x^n - 1)$. Because of what we know about ideals in $GF(q)[x]/(x^n - 1)$ we can immediately make some observations about cyclic codes:

- A cyclic code has a generator polynomial $g(x)$, which is the generator of the ideal. Let the degree of g be r, where $r < n$.

- Every code polynomial in the code can be expressed as a multiple of the generator.
$$c(x) = m(x)g(x),$$
where $m(x)$ is the *message polynomial*. The degree of m is less than $n - r$.

- The generator is a factor of $x^n - 1$ in $GF(q)[x]$.

Example 1 We will consider cyclic codes of length 15 with binary coefficients. We need to find the factors of $x^n - 1$ in some field. Observe that

$$15|2^4 - 1,$$

so we are dealing in the field $GF(16)$. The conjugacy classes in $GF(16)$ are

$$\begin{aligned}\{1\} &\leftrightarrow x+1\\ \{\alpha,\alpha^2,\alpha^4,\alpha^8\} &\leftrightarrow 1+x+x^4\\ \{\alpha^3,\alpha^6,\alpha^9,\alpha^{12}\} &\leftrightarrow 1+x+x^2+x^3+x^4\\ \{\alpha^5,\alpha^{10}\} &\leftrightarrow 1+x+x^2\\ \{\alpha^7,\alpha^{14},\alpha^{13},\alpha^{11}\} &\leftrightarrow 1+x^3+x^4\end{aligned}$$

Thus

$$x^{15} - 1 = (x+1)(1+x+x^4)(1+x+x^2+x^3+x^4)(1+x+x^2)(1+x^3+x^4)$$

So: degrees 1,2,4,4,4. If we want a generator of, say, degree 9, we could take

$$g(x) = (x+1)(1+x+x^4)(1+x+x^2+x^3+x^4)$$

If we want a generator of degree 5 we could take

$$g(x) = (x+1)(1+x+x^4)$$

or

$$g(x) = (x+1)(1+x+x^2+x^3+x^4)$$

In fact, in this case, we can get generator polynomials of any degree from 1 to 15. So we have codes

$$(15,0)(15,1),\ldots,(15,15)$$

□

A message sequence $(m_0, m_1, \ldots, m_{k-1})$ (where $k = n - r$) corresponds to a message polynomial

$$m(x) = m_0 + \cdots + m_{k-1}x^{k-1}.$$

Then the message polynomial corresponding to m is

$$c_m(x) = m(x)g(x).$$

We can write this as

$$c_m(x) = [m_0, m_1, \ldots, m_{k-1}]\begin{bmatrix} g(x)\\ xg(x)\\ \vdots\\ x^{k-1}g(x)\end{bmatrix}$$

Taking the next step, we can go back to a matrix representation,

$$\mathbf{c}_m = [m_0, m_1, \ldots, m_{k-1}] \begin{bmatrix} g_0 & g_1 & \cdots & g_r & & & & & \\ & g_0 & g_1 & \cdots & g_r & & & & \\ & & g_0 & g_1 & \cdots & g_r & & & \\ & & & \ddots & \ddots & & \ddots & & \\ & & & & g_0 & g_1 & \cdots & g_r & \\ & & & & & g_0 & g_1 & \cdots & g_r \end{bmatrix} = \mathbf{m}G$$

So we have a linear code, and can write the generator matrix corresponding to it.
Note: G is $k \times n$.

Let $h(x)$ be **parity check polynomial**, that is a polynomial such that

$$x^n - 1 = g(x)h(x).$$

Since codewords are multiples of $g(x)$, then for a codeword,

$$c(x)h(x) = m(x)g(x)h(x) = m(x)(x^n - 1) \equiv 0 \pmod{x^n - 1}.$$

We let

$$s(x) = c(x)h(x) \pmod{x^n - 1}.$$

be the **syndrome polynomial**. If $s(x)$ is identically zero, then $c(x)$ is a codeword.
Now let's put this in matrix form.

$$s(x) = c(x)h(x) = \sum_{i=0}^{n-1} c_i x^i \sum_{j=0}^{n-1} h_j x^j \pmod{x^n - 1}.$$

Performing the multiplication,

$$s_k = \sum_{i=0}^{n-1} c_i h_{((k-i))_n} \qquad k = 0, 1, \ldots, n-1.$$

Writing the last $n-k$ of these out, we have

$$[s_k, s_{k+1}, \ldots, s_{n-1}] = [c_0, c_1, \ldots, c_{n-1}] \begin{bmatrix} h_k & h_{k-1} & \cdots & h_1 & h_0 & & & & & \\ & h_k & h_{k-1} & \cdots & h_1 & h_0 & & & & \\ & & \ddots & \ddots & & & & & & \\ & & & & h_k & h_{k-1} & \cdots & h_1 & h_0 & \\ & & & & & h_k & h_{k-1} & \cdots & h_1 & h_0 \end{bmatrix}^T = \mathbf{c}H^T.$$

LINEAR BLOCK CODES.

Error-Control Coding

Error-control coding techniques are used to detect and/or correct errors that occur in the message transmission in a digital communication system. The transmitting side of the error-control coding adds redundant bits or symbols to the original information signal sequence. The receiving side of the error-control coding uses these redundant bits or symbols to detect and/or correct the errors that occurred during transmission. The transmission coding process is known as *encoding*, and the receiving coding process is known as *decoding*.

There are two major classes in error-control code: block and convolutional. In block coding, successive blocks of *K* information (message) symbols are formed.

The coding algorithm then transforms each block into a codeword consisting of *n* symbols where *n*>*k*. This structure is called an (*n*,*k*) code. The ratio *k/n* is called the code rate. A key point is that each codeword is formed independently from other codewords.

An error-control code is a *linear code* if the transmitted signals are a linear function of the information symbols. The code is called a *systematic code* if the information symbols are transmitted without being altered. Most block codes are systematic, whereas most convolutional codes are nonsystematic.

Almost all codes used for error control are linear. The symbols in a code can be either binary or non-binary. Binary symbols are the familiar '0' and '1'.

Linear Block Codes

Linear block coding is a generic coding method. Other coding methods, such as Hamming and BCH codes, are special cases of linear block coding. The codeword vector of a linear block code is a linear mapping of the message vector. The codeword ***x*** and the message ***m*** have the relationship

$$\mathbf{x} = \mathbf{mG}$$

where $\mathbf{G}$ is a *K*-by-*N* matrix and is known as the *generator matrix*.

Linear block code is called a systematic linear code if the generator matrix has the form

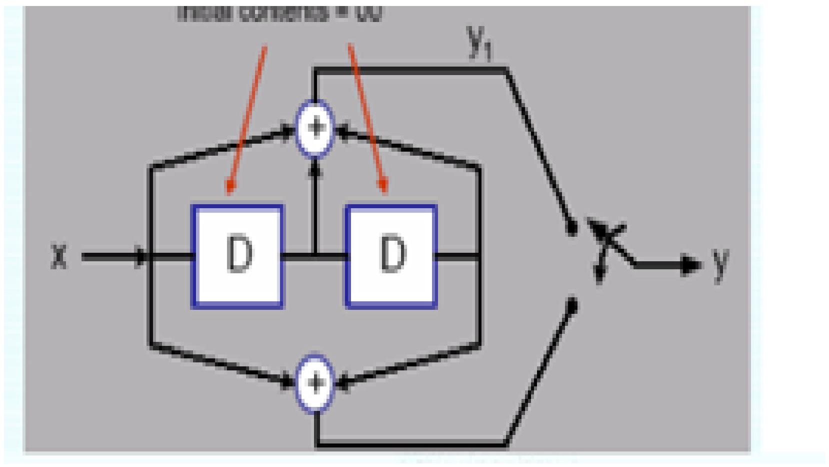
$$\mathbf{G} = [\mathbf{P} \vdots \mathbf{I}_k]$$

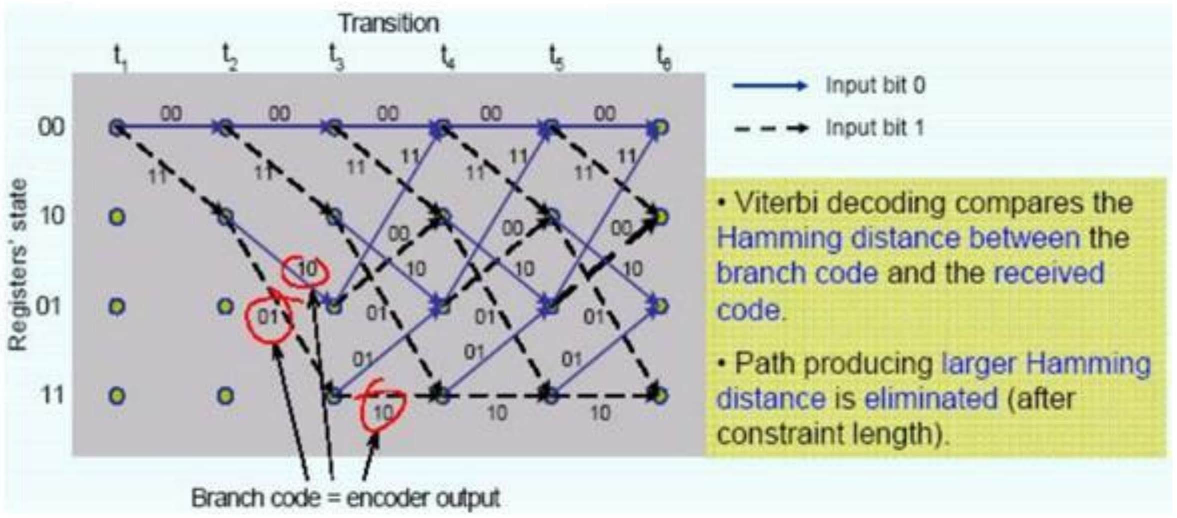
where $\mathbf{P}$ is an *(n-k)-by-k* matrix and $\mathbf{I}_k$ is a *k-by-k* identity matrix. A systematic linear code renders a length *k* message into a length *n* codeword where the last *k* bits are exactly the original message and the first *(n-k)* bits are redundant. These redundant bits serve as parity-check digits.

VITERBI ALGORITHM.

ML algorithm is too complex to search all available pathes.

- End to end calculation.
- Viterbi algorithm performs ML decoding by reducing its complexity.
- Eliminate least likely trellis path at each transmission stage.
- Reduce decoding complexity with early rejection of unlike pathes.
- Viterbi algorithm gets its efficiency via concentrating on suvival paths of the trellis.




Transition
t_1 t_2 t_3 t_4 t_5 t_6
Registers' state
00
10
01
11
Input bit 0
Input bit 1
Branch code = encoder output

• Viterbi decoding compares the Hamming distance between the branch code and the received code.

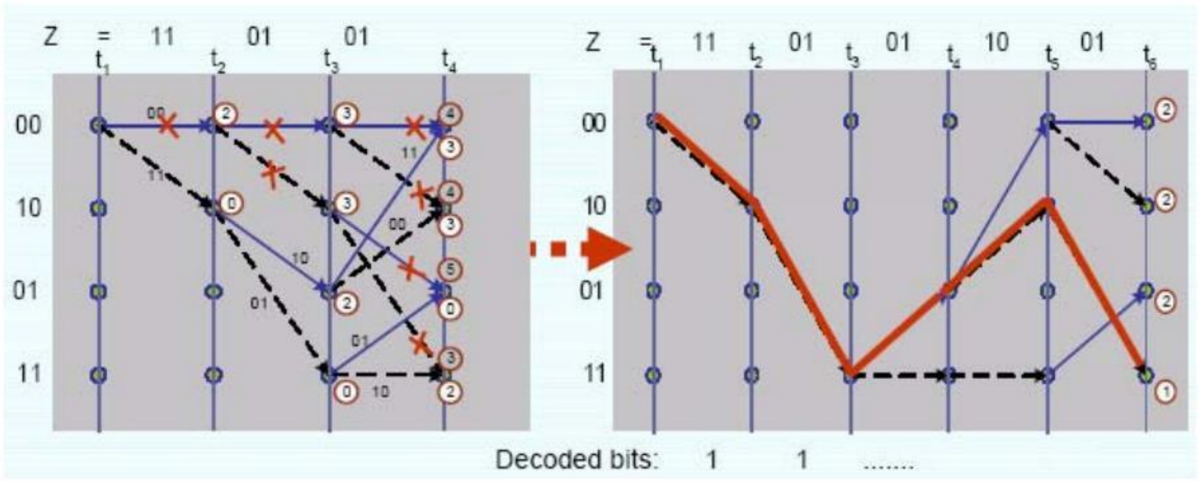
• Path producing larger Hamming distance is eliminated (after constraint length).

Example of viterbi Decoding:

Input data: m =1 1 0 1 1

Codeword: X = 11 01 01 00 01

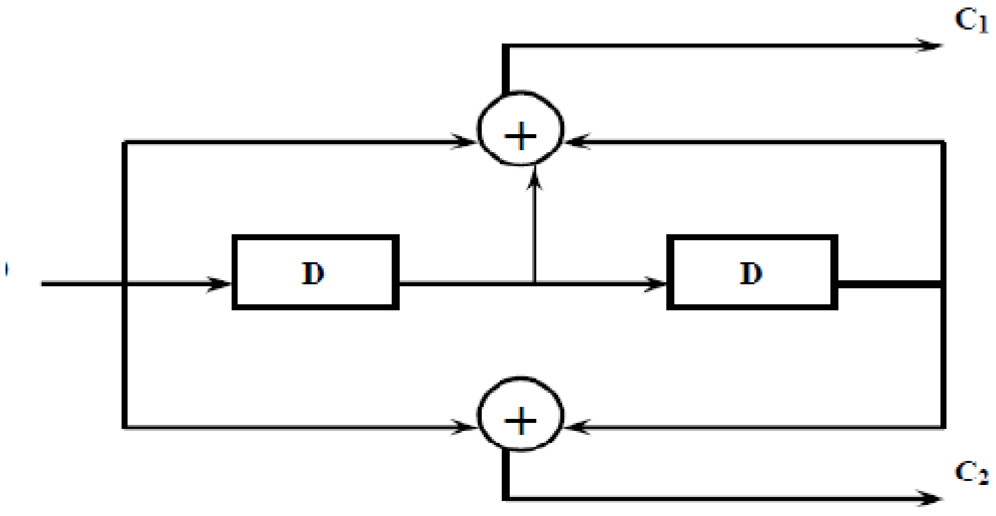
Received code: Z = 11 01 01 10 01



CONVOLUTIONAL CODES.

* Convolutional codes are widely used as channel codes in practical communication systems for error correction.
* The encoded bits depend on the current k input bits and a few past input bits.
* The main decoding strategy for convolutional codes is based on the widely used Viterbi algorithm.
* Convolutional codes are commonly described using two parameters: the code rate and the constraint length. The code rate, k/n, is expressed as a ratio of the number of bits into the convolutional encoder (k) to the number of channel symbols output by the convolutional encoder (n) in a given encoder cycle.
* The constraint length parameter, K, denotes the "length" of the convolutional encoder, i.e. how many k-bit stages are available to feed the combinatorial logic that produces the output symbols. Closely related to K is the parameter m, which can be thought of as the memory length of the encoder. A simple convolutional encoder is shown below(fig 3.1).

The information bits are fed in small groups of k-bits at a time to a shift register. The output encoded bits are obtained by modulo-2 addition (EXCLUSIVE-OR operation) of the input information bits and the contents of the shift registers which are a few previous information bits.



A convolutional encoder with k=1, n=2 and r=1/2

The operation of a convolutional encoder can be explained in several but equivalent ways such as, by

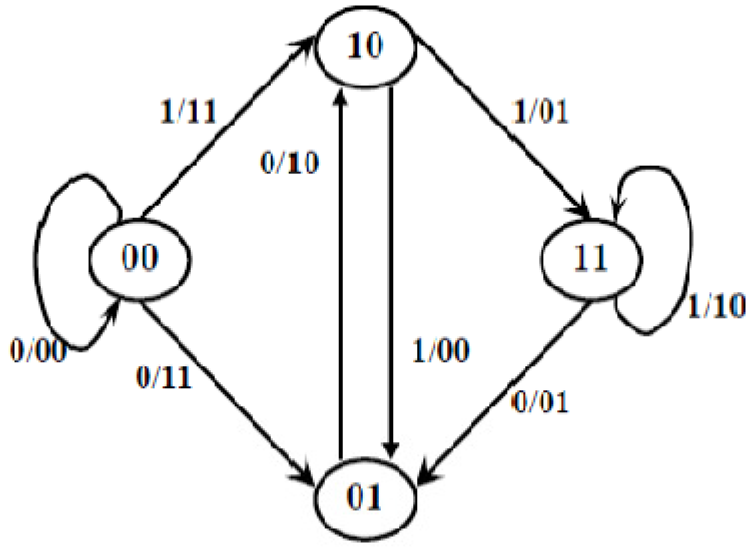
a) state diagram representation. b) tree diagramrepresentation.

c) trellis diagram representation.

a) **State Diagram Representation:** A convolutional encoder may be defined as a finite state machine. Contents of the rightmost (K-1) shift register stages define the states of the encoder. So, the encoder in **Fig. 3.1** has four states.

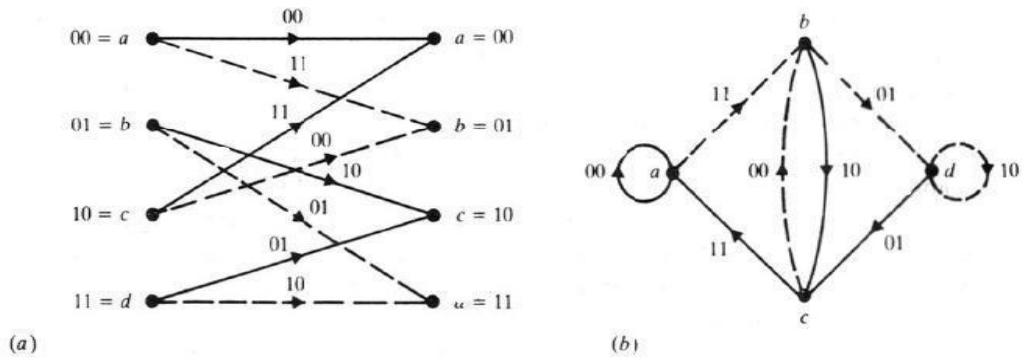
The transition of an encoder from one state to another, as caused by input bits, is depicted in the state diagram.**Fig. 3.2** shows the state diagram of the encoder in **Fig. 3.1**.

A new input bit causes a transition from one state to another.



State diagram representation for the encoder.

***b)* Tree Diagram Representation:** The tree diagram representation shows all possible information and encoded sequences for the convolutional encoder. **Fig. 3.3** shows the tree diagram for the encoder in **Fig. 3.1**. The encoded bits are labeled on the branches of the tree. Given an nput sequence, the encoded sequence can be directly read from the tree. **Representing convolutional codes compactly: code trellis and state diagram:** State diagram



(a) (b)

Input	1	1	0	1	1	1	0	0	1	0	0	0	
State	a	b	d	c	b	d	d	c	a	b	c	a	a
Output	11	01	01	00	01	10	01	11	11	10	11	00	

(c)

Inspecting state diagram: Structural properties of convolutional codes:

• Each new block of *k* input bits causes a transition into new state

• Hence there are 2*k* branches leaving each state

• Assuming encoder zero initial state, encoded word for any input of *k* bits can thus be obtained. For instance, below for **u**=(1 1 1 0 1), encoded word **v**=(1 1, 1 0, 0 1, 0 1, 1 1, 1 0, 1 1, 1 1) is produced:

- encoder state diagram for (*n*,*k*,*L*)=(2,1,2) code - note that the number of states is 2*L*+1 = 8

Distance for some convolutional codes:

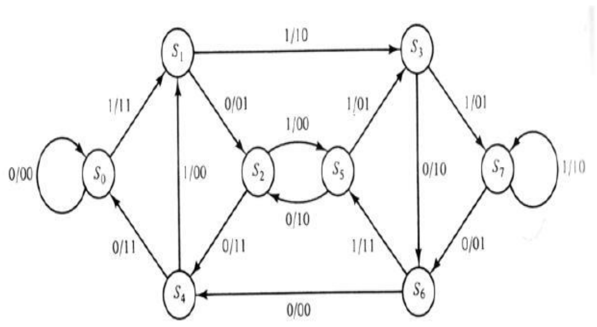


Fig. A tree diagram for the encoder

c) **Trellis Diagram Representation:**

The trellis diagram of a convolutional code is obtained from its state diagram. All state transitions at each time step are explicitly shown in the diagram to retain the time dimension, as is present in the corresponding tree diagram.

Usually, supporting descriptions on state transitions, corresponding input and output bits etc. are labeled in the trellis diagram.

It is interesting to note that the trellis diagram, which describes the operation of the encoder, is very convenient for describing the behavior of the corresponding decoder, especially when the famous „Viterbi Algorithm (VA)" is followed.

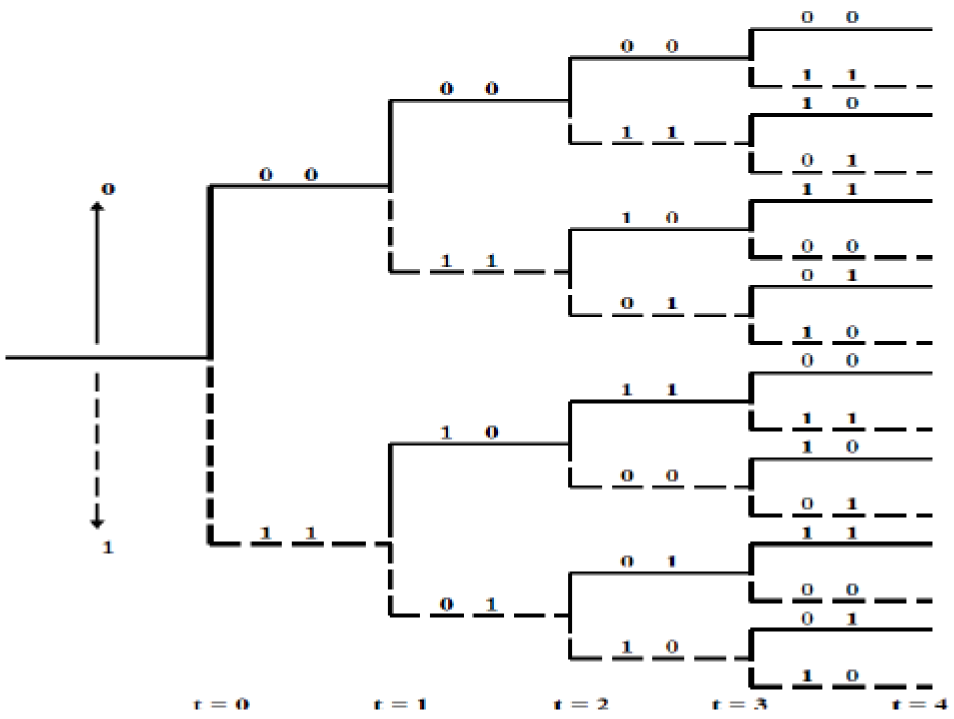


Fig.. Trellis diagram for the encoder in Fig. 3.1

1/3 rate convolutional encoder has the following generator g1=(1 0 0), g2=(1 0 1), g3=(1 1 1)

i)Draw the encoder circuit corresponding to this code (3)

ii) Draw the code tree (3) iii) Draw the state Diagram (3) v) Draw the trellis Diagram (3)

v)This code is used for transmission over a Awgn channel with hard decision decoder. Using viterbi algorithm