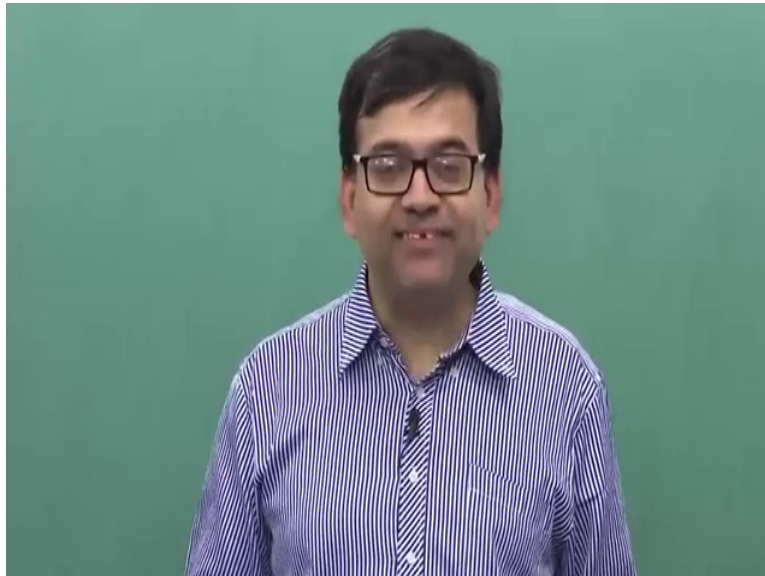


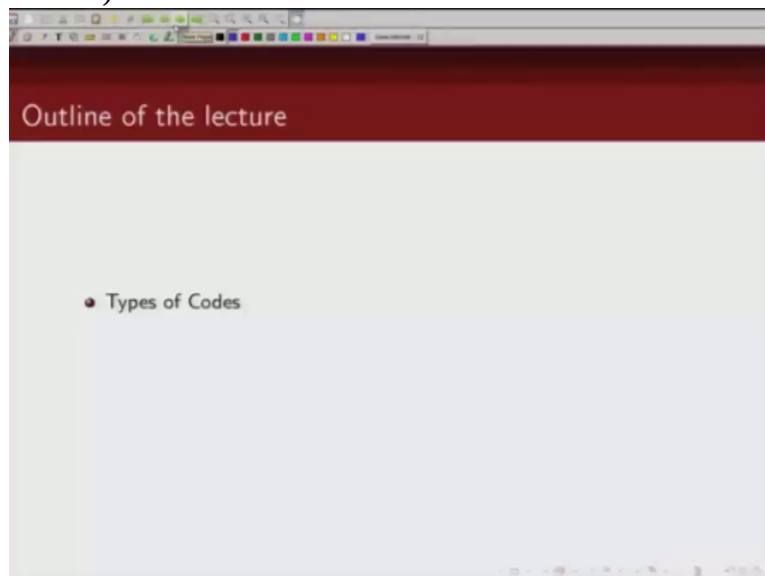
**An Introduction to Coding Theory**  
**Professor Adrish Banerji**  
**Department of Electrical Engineering**  
**Indian Institute of Technology, Kanpur**  
**Module 01**  
**Lecture Number 03**  
**Introduction to Error Control Coding-III**

(Refer Slide Time 00:13)



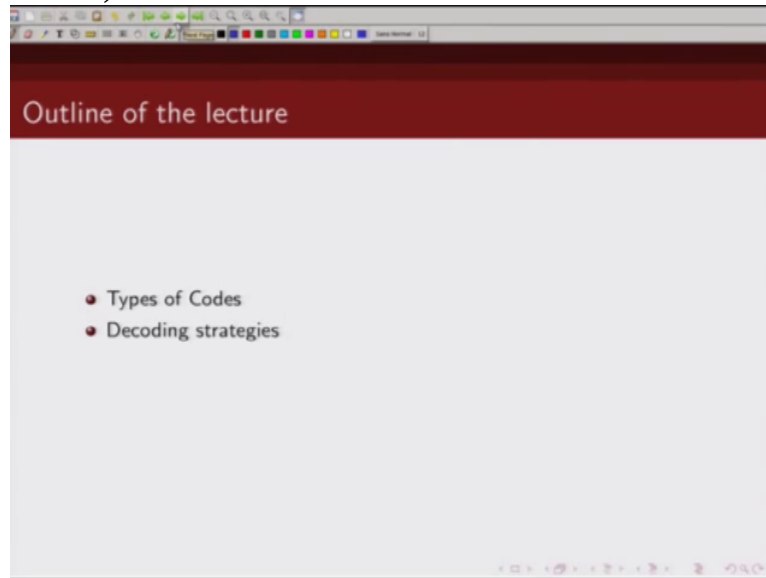
In this lecture we are going to conclude our discussion on introduction and so, in this lecture I will first describe what is a difference between block codes and convolutional codes. And then we will talk about very simple decoding strategies and finally we will explain by what we mean by forward error correction, automatic repeat request and hybrid a r q. So as I said,

(Refer Slide Time 00:42)



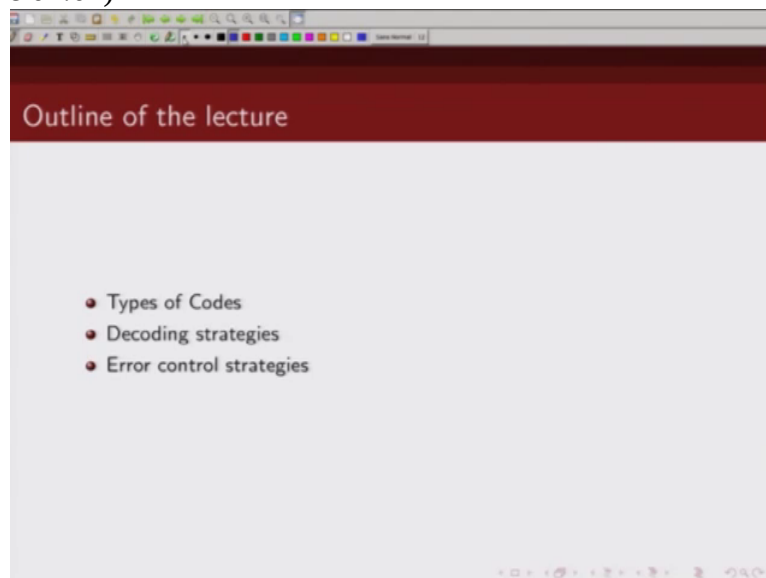
we will first describe, so error correcting codes can be broadly classified into two classes, block codes and convolutional codes. We will describe what is meant by block code and what is meant by convolutional code and we will bring out the difference and the similarities between the two. Then we will talk about

(Refer Slide Time 01:04)



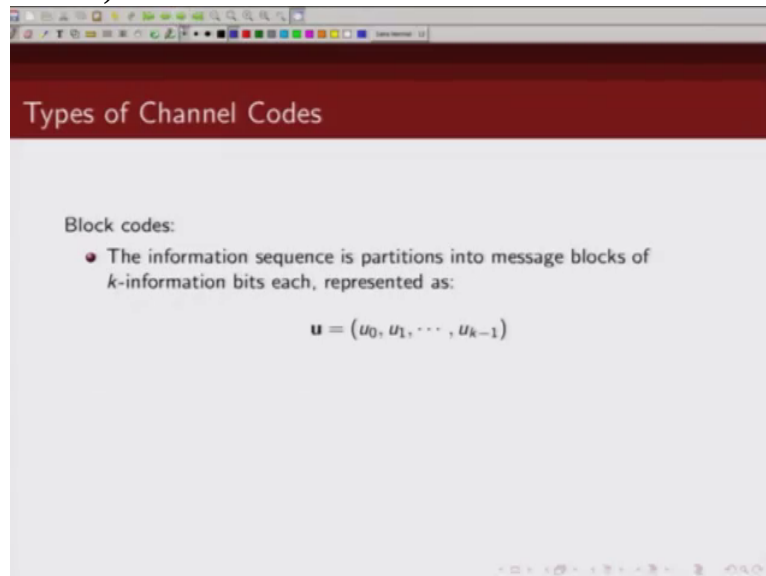
various decoding strategies and finally

(Refer Slide Time 01:07)



we will talk about what we mean by forward error correction, hybrid ARQ and automatic repeat request.

(Refer Slide Time 01:17)



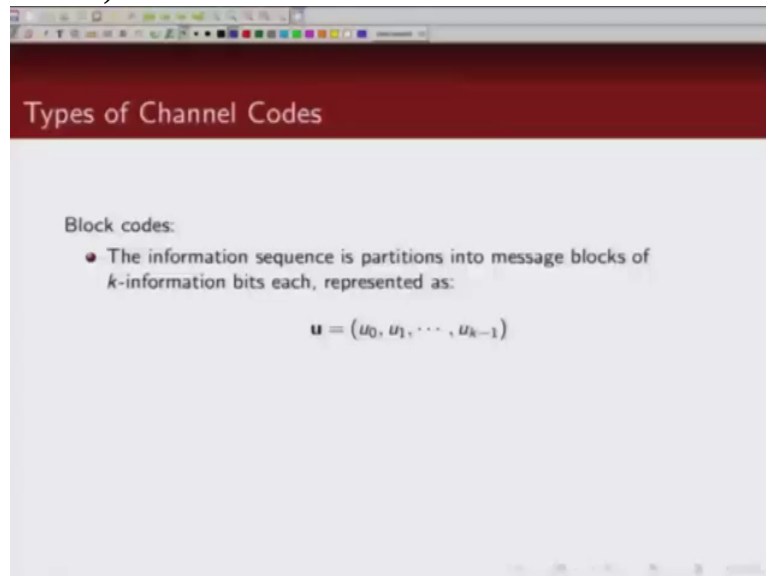
So we will start with what is block codes. So as the name suggests, in block codes we

(Refer Slide Time 01:25)



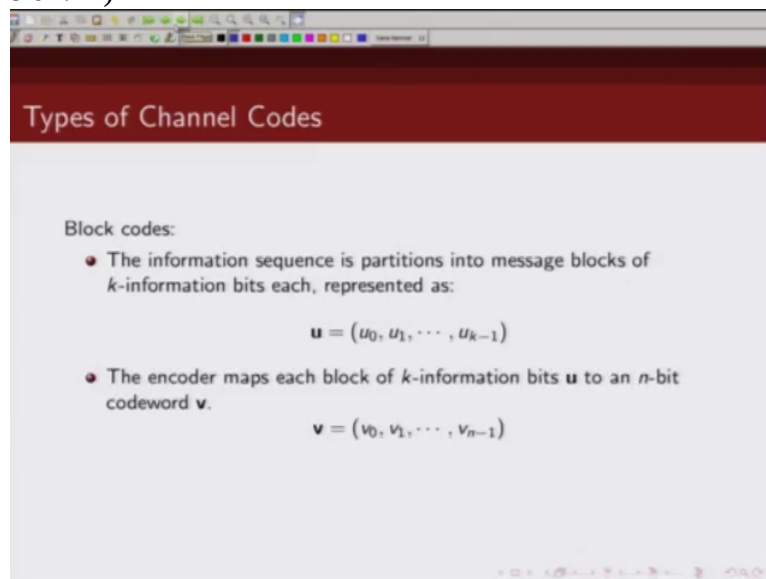
take a block of  $k$ -bits and map it to an  $n$ -bit codeword. So our informative sequence is parsed into

(Refer Slide Time 01:37)



blocks of  $k$ -bits And we take this block of  $k$ -bits and map it to block of  $n$ -bits. So

(Refer Slide Time 01:47)



we denote our information sequence by  $\mathbf{u}$ . So this is a  $k$ -bit sequence  $u_0, u_1$  to  $u_{k-1}$  and our encoder is going to map this  $k$ -bits into a  $n$ -bit sequence which is denoted by  $\mathbf{v}$ . Now

(Refer Slide Time 02:07)

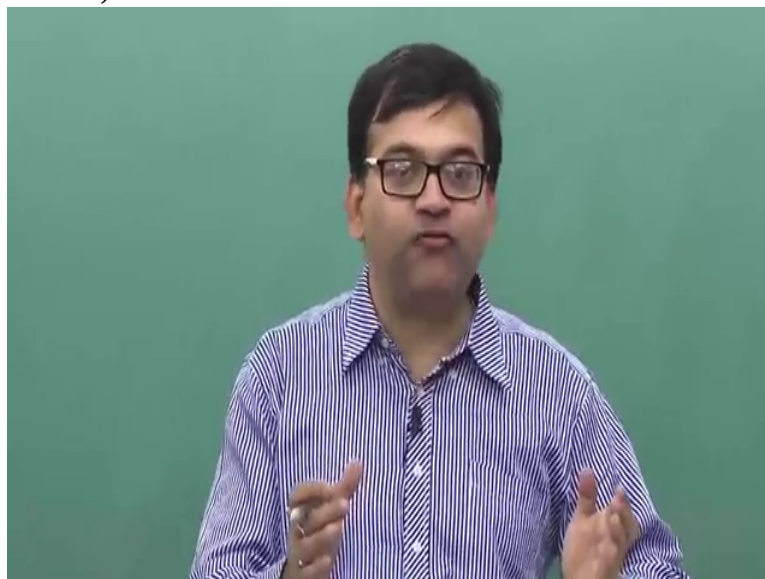
Types of Channel Codes

Block codes:

- The information sequence is partitioned into message blocks of  $k$ -information bits each, represented as:  
$$\mathbf{u} = (u_0, u_1, \dots, u_{k-1})$$
- The encoder maps each block of  $k$ -information bits  $\mathbf{u}$  to an  $n$ -bit codeword  $\mathbf{v}$ .  
$$\mathbf{v} = (v_0, v_1, \dots, v_{n-1})$$
- The encoder for a block code is memoryless.

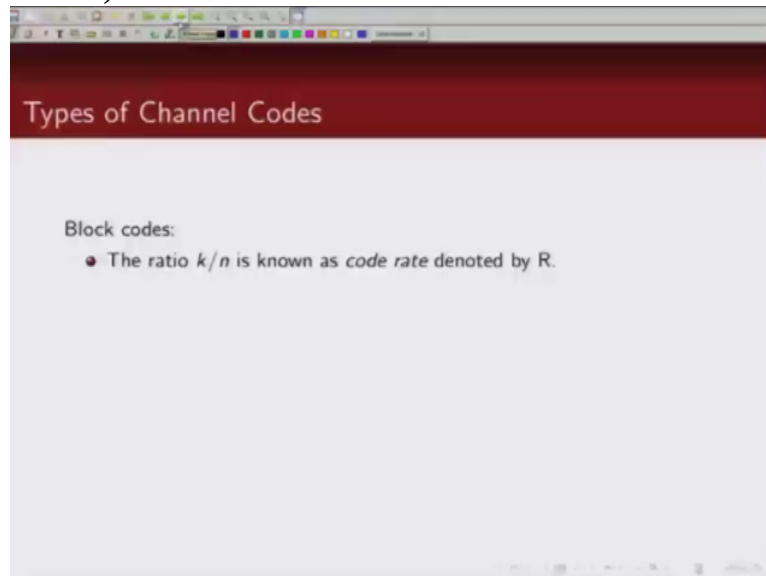
in block code, the encoder is memoryless. What do we mean by that? So when we encode a block of  $k$ -bits,

(Refer Slide Time 02:15)



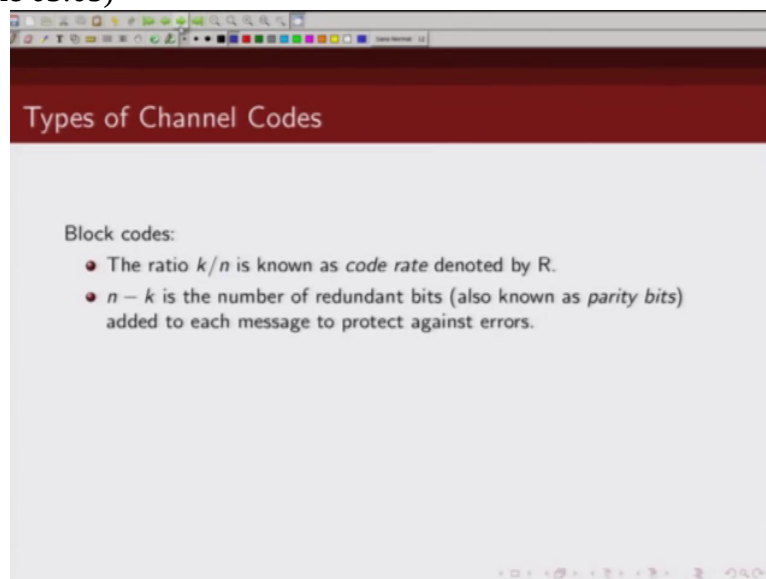
our output depends only on that current block of  $k$ -bits. It does not depend on what was the previous blocks of data. It only depends; output only depends on the current  $k$ -bits. So that is one property of block codes which makes it different from convolutional codes. Block codes are memoryless.

(Refer Slide Time 02:43)



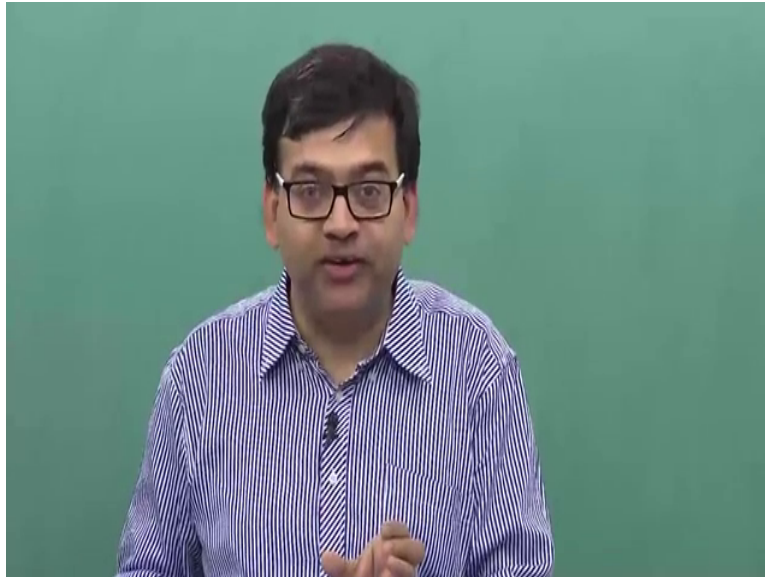
As we mentioned in the previous lectures, we defined our code rate to be the ratio of number of information bits to number of coded bits. So the ratio of information bits to coded bit is basically, will be denoted by code rate. And

(Refer Slide Time 03:05)



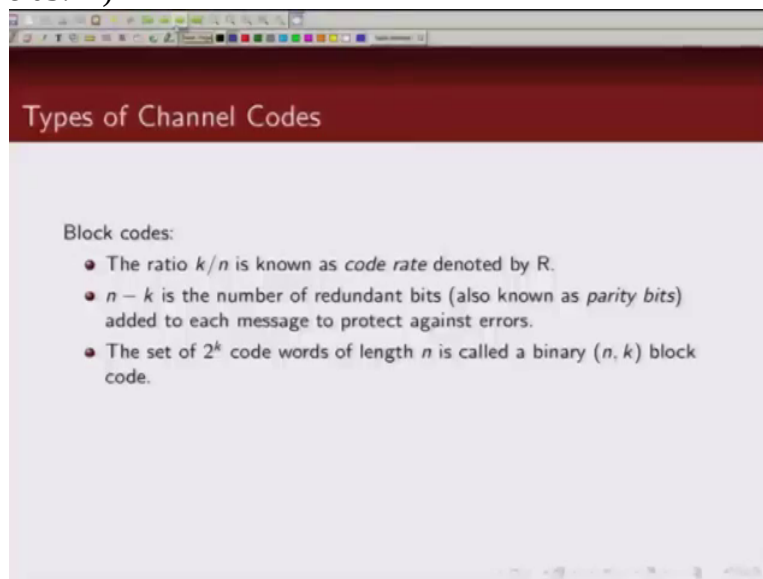
it is typically denoted by  $R$ .  $k$  is number of information bits.  $n$  is number of coded bits. So  $n$  minus  $k$  is number of redundant bits that we are adding to our information bits

(Refer Slide Time 03:19)



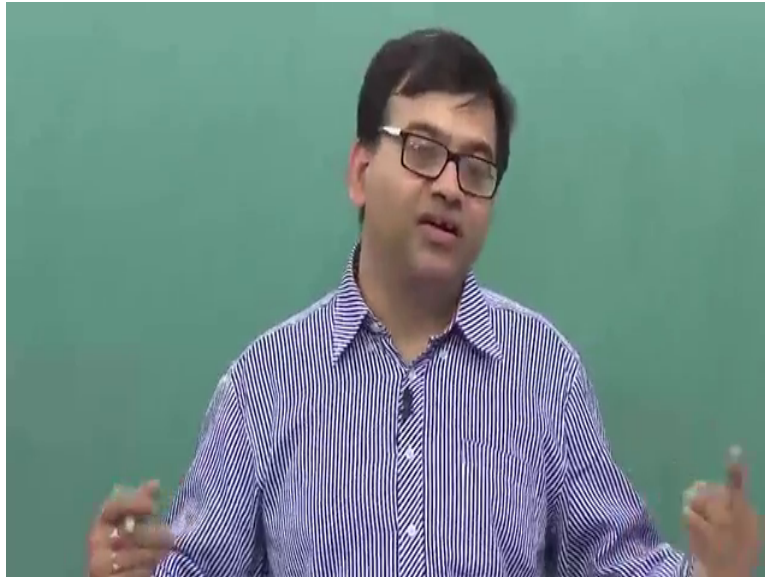
and these are also known as parity bits. If you are

(Refer Slide Time 03:24)



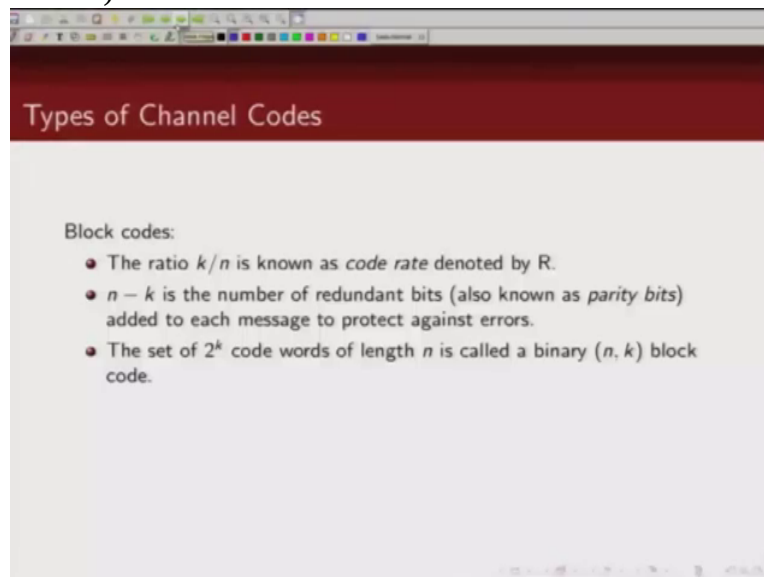
considering, without loss of generality, we will basically consider in this set of lectures binary codewords so our information sequence

(Refer Slide Time 03:34)



consists of zeros and ones, similarly our code sequence also consists of zeros and ones. Since we are considering the block of  $k$ -bits and binary codewords, the number of codewords is basically 2 raised to power  $k$ . So a binary

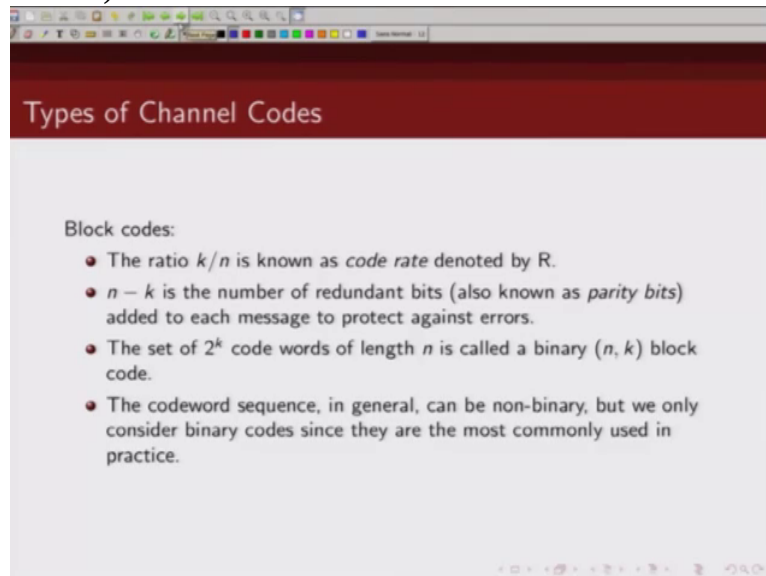
(Refer Slide Time 03:52)



$n$   $k$  block code consists of  $2^k$  codewords each of length  $n$ . Now these

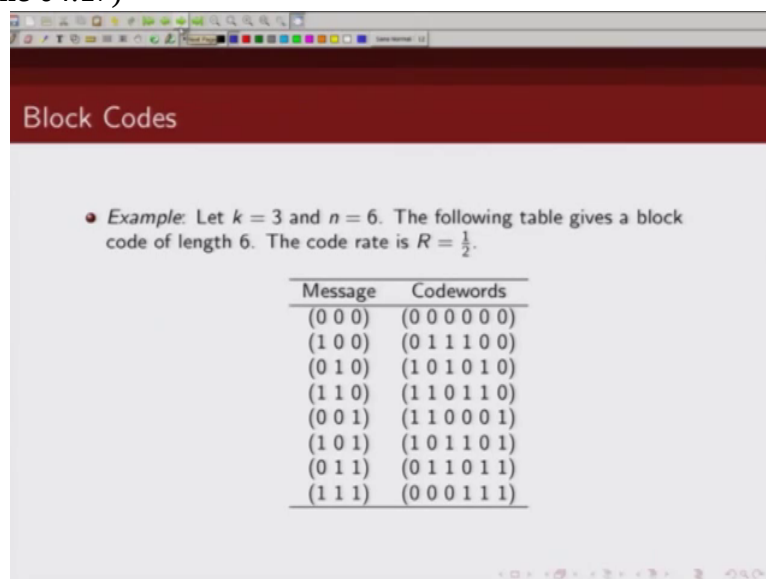


(Refer Slide Time 04:00)



codewords need not be binary, however it's the same theory mostly applies to non-binary codewords as well so we will restrict our discussion to binary codewords.

(Refer Slide Time 04:17)



So let us consider an example of linear block code. So in this example, a number of information

(Refer Slide Time 04:25)

Block Codes

- Example: Let  $k = 3$  and  $n = 6$ . The following table gives a block code of length 6. The code rate is  $R = \frac{1}{2}$ .

Message	Codewords
(0 0 0)	(0 0 0 0 0 0)
(1 0 0)	(0 1 1 1 0 0)
(0 1 0)	(1 0 1 0 1 0)
(1 1 0)	(1 1 0 1 1 0)
(0 0 1)	(1 1 0 0 0 1)
(1 0 1)	(1 0 1 1 0 1)
(0 1 1)	(0 1 1 0 1 1)
(1 1 1)	(0 0 0 1 1 1)

bits is 3, the number of coded bits is 6.

(Refer Slide Time 04:29)

Block Codes

- Example: Let  $k = 3$  and  $n = 6$ . The following table gives a block code of length 6. The code rate is  $R = \frac{1}{2}$ .

Message	Codewords
(0 0 0)	(0 0 0 0 0 0)
(1 0 0)	(0 1 1 1 0 0)
(0 1 0)	(1 0 1 0 1 0)
(1 1 0)	(1 1 0 1 1 0)
(0 0 1)	(1 1 0 0 0 1)
(1 0 1)	(1 0 1 1 0 1)
(0 1 1)	(0 1 1 0 1 1)
(1 1 1)	(0 0 0 1 1 1)

So the code rate which is ratio of information bits to number of coded bits is 3 by 6 which is half.

(Refer Slide Time 04:38)

Block Codes

- Example: Let  $k = 3$  and  $n = 6$ . The following table gives a block code of length 6. The code rate is  $R = \frac{1}{2}$ .

Message	Codewords
(0 0 0)	(0 0 0 0 0 0)
(1 0 0)	(0 1 1 1 0 0)
(0 1 0)	(1 0 1 0 1 0)
(1 1 0)	(1 1 0 1 1 0)
(0 0 1)	(1 1 0 0 0 1)
(1 0 1)	(1 0 1 1 0 1)
(0 1 1)	(0 1 1 0 1 1)
(1 1 1)	(0 0 0 1 1 1)

So what we have here is basically our message bits. Now  $k$  is 3, that means there are 2 raised to power 3, which is 8 codewords and these are basically from 0 0 0 to 1 1 1, these are the 8 codewords. Now the message, these are the 8 message bits and corresponding to these message bits, these are the 8 codewords. Now 0 0 0 is mapped to all zero sequence, 1 0 0 is mapped to 0 1 1 1 0 0, likewise other sequences have been mapped. So let us look at how we have mapped, how have we found out the message parity bits for this particular codeword. So let's look at each of the columns of these codewords. So let us look at this column first

(Refer Slide Time 05:35)

Block Codes

- Example: Let  $k = 3$  and  $n = 6$ . The following table gives a block code of length 6. The code rate is  $R = \frac{1}{2}$ .

Message	Codewords
(0 0 0)	(0 0 0 0 0 0)
(1 0 0)	(0 1 1 1 0 0)
(0 1 0)	(1 0 1 0 1 0)
(1 1 0)	(1 1 0 1 1 0)
(0 0 1)	(1 1 0 0 0 1)
(1 0 1)	(1 0 1 1 0 1)
(0 1 1)	(0 1 1 0 1 1)
(1 1 1)	(0 0 0 1 1 1)

which is 0 0 0 0 1 1 1 1. So how was this column, how did we map to get this column? If you look at information bits, this column is nothing but same as

(Refer Slide Time 05:54)

Block Codes

- Example: Let  $k = 3$  and  $n = 6$ . The following table gives a block code of length 6. The code rate is  $R = \frac{1}{2}$ .

Message	Codewords
(0 0 0)	(0 0 0 0 0 0)
(1 0 0)	(0 1 1 1 0 0)
(0 1 0)	(1 0 1 0 1 0)
(1 1 0)	(1 1 0 1 1 0)
(0 0 1)	(1 1 0 0 0 1)
(1 0 1)	(1 0 1 1 0 1)
(0 1 1)	(0 1 1 0 1 1)
(1 1 1)	(0 0 0 1 1 1)

this information bit. You can see 0 0 0 0 1 1 1 1 1. Similarly look at this one. This

(Refer Slide Time 06:08)

Block Codes

- Example: Let  $k = 3$  and  $n = 6$ . The following table gives a block code of length 6. The code rate is  $R = \frac{1}{2}$ .

Message	Codewords
(0 0 0)	(0 0 0 0 0 0)
(1 0 0)	(0 1 1 1 0 0)
(0 1 0)	(1 0 1 0 1 0)
(1 1 0)	(1 1 0 1 1 0)
(0 0 1)	(1 1 0 0 0 1)
(1 0 1)	(1 0 1 1 0 1)
(0 1 1)	(0 1 1 0 1 1)
(1 1 1)	(0 0 0 1 1 1)

column is same as this column,

(Refer Slide Time 06:12)

Block Codes

- Example: Let  $k = 3$  and  $n = 6$ . The following table gives a block code of length 6. The code rate is  $R = \frac{1}{2}$ .

Message	Codewords
(0 0 0)	(0 0 0 0 0 0)
(1 0 0)	(0 1 1 1 0 0)
(0 1 0)	(1 0 1 0 1 0)
(1 1 0)	(1 1 0 1 1 0)
(0 0 1)	(1 1 0 0 0 1)
(1 0 1)	(1 0 1 1 0 1)
(0 1 1)	(0 1 1 0 1 1)
(1 1 1)	(0 0 0 1 1 1)

0 0 1 1 0 0 1 1 and this column is

(Refer Slide Time 06:22)

Block Codes

- Example: Let  $k = 3$  and  $n = 6$ . The following table gives a block code of length 6. The code rate is  $R = \frac{1}{2}$ .

Message	Codewords
(0 0 0)	(0 0 0 0 0 0)
(1 0 0)	(0 1 1 1 0 0)
(0 1 0)	(1 0 1 0 1 0)
(1 1 0)	(1 1 0 1 1 0)
(0 0 1)	(1 1 0 0 0 1)
(1 0 1)	(1 0 1 1 0 1)
(0 1 1)	(0 1 1 0 1 1)
(1 1 1)	(0 0 0 1 1 1)

same as this column

(Refer Slide Time 06:27)

Block Codes

- Example: Let  $k = 3$  and  $n = 6$ . The following table gives a block code of length 6. The code rate is  $R = \frac{1}{2}$ .

Message	Codewords
(0 0 0)	(0 0 0 0 0 0)
(1 0 0)	(0 1 1 1 0 0)
(0 1 0)	(1 0 1 0 1 0)
(1 1 0)	(1 1 0 1 1 0)
(0 0 1)	(1 1 0 0 0 1)
(1 0 1)	(1 0 1 1 0 1)
(0 1 1)	(0 1 1 0 1 1)
(1 1 1)	(0 0 0 1 1 1)

So in other words, this bit of the codeword is same as this bit of the information sequence. This bit of the codeword is same as this bit of the information sequence. This bit of the codeword is same as this bit of the information sequence. Now let's look at this one.

(Refer Slide Time 06:50)

Block Codes

- Example: Let  $k = 3$  and  $n = 6$ . The following table gives a block code of length 6. The code rate is  $R = \frac{1}{2}$ .

Message	Codewords
(0 0 0)	(0 0 0 0 0 0)
(1 0 0)	(0 1 1 1 0 0)
(0 1 0)	(1 0 1 0 1 0)
(1 1 0)	(1 1 0 1 1 0)
(0 0 1)	(1 1 0 0 0 1)
(1 0 1)	(1 0 1 1 0 1)
(0 1 1)	(0 1 1 0 1 1)
(1 1 1)	(0 0 0 1 1 1)

So if we do a xor of these two, look at this  $x \oplus 0$  plus 0 is 0, 1 plus 0 is 1, 0 plus 1 is 1, 1 plus 1 is 0. We are talking about binary, addition over binary field so 0 plus 0 is 0, 0 plus 1 is 1, 1 plus 0 is 1, and 1 plus 1 is 0, it is modulo two addition. So 1 plus 1 is 0, this is 0 plus 0 is 0, 1 plus 0 is 1, 0 plus 1 is 1 and 1 plus 1 is 0. So if we, let's say, denote this by  $u_0, u_1, u_2$  and we denote

(Refer Slide Time 07:46)

**Block Codes**

- Example: Let  $k = 3$  and  $n = 6$ . The following table gives a block code of length 6. The code rate is  $R = \frac{1}{2}$ .

Message	Codewords
(0 0 0)	(0 0 0 0 0 0)
(1 0 0)	(0 1 1 1 0 0)
(0 1 0)	(1 0 1 0 1 0)
(1 1 0)	(1 1 0 1 1 0)
(0 0 1)	(1 1 0 0 0 1)
(1 0 1)	(1 0 1 1 0 1)
(0 1 1)	(0 1 1 0 1 1)
(1 1 1)	(0 0 0 1 1 1)

$u_2 \ u_1 \ u_0$

this by  $v_0, v_1, v_2, v_3, v_4$  and  $v_5$ , what we have

(Refer Slide Time 07:55)

**Block Codes**

- Example: Let  $k = 3$  and  $n = 6$ . The following table gives a block code of length 6. The code rate is  $R = \frac{1}{2}$ .

Message	Codewords
(0 0 0)	(0 0 0 0 0 0)
(1 0 0)	(0 1 1 1 0 0)
(0 1 0)	(1 0 1 0 1 0)
(1 1 0)	(1 1 0 1 1 0)
(0 0 1)	(1 1 0 0 0 1)
(1 0 1)	(1 0 1 1 0 1)
(0 1 1)	(0 1 1 0 1 1)
(1 1 1)	(0 0 0 1 1 1)

$u_2 \ u_1 \ u_0$        $v_0 \ v_1 \ v_2 \ v_3 \ v_4 \ v_5$

found out so far is  $v_5$  is same as  $u_2$ ,  $v_4$  is same as  $u_1$ ,

(Refer Slide Time 08:08)

Block Codes

- Example: Let  $k = 3$  and  $n = 6$ . The following table gives a block code of length 6. The code rate is  $R = \frac{1}{2}$ .

Message	Codewords
(0 0 0)	(0 0 0 0 0 0)
(1 0 0)	(0 1 1 1 0 0)
(0 1 0)	(1 0 1 0 1 0)
(1 1 0)	(1 1 0 1 1 0)
(0 0 1)	(1 1 0 0 0 1)
(1 0 1)	(1 0 1 1 0 1)
(0 1 1)	(0 1 1 0 1 1)
(1 1 1)	(0 0 0 1 1 1)

$v_5 = u_2$   
 $v_4 = u_1$

$u_6 \ u_1 \ u_2 \quad v_6 \ v_1 \ v_2 \ v_3 \ v_4 \ v_5$

$v_3$  is same as  $u_0$

(Refer Slide Time 08:13)

Block Codes

- Example: Let  $k = 3$  and  $n = 6$ . The following table gives a block code of length 6. The code rate is  $R = \frac{1}{2}$ .

Message	Codewords
(0 0 0)	(0 0 0 0 0 0)
(1 0 0)	(0 1 1 1 0 0)
(0 1 0)	(1 0 1 0 1 0)
(1 1 0)	(1 1 0 1 1 0)
(0 0 1)	(1 1 0 0 0 1)
(1 0 1)	(1 0 1 1 0 1)
(0 1 1)	(0 1 1 0 1 1)
(1 1 1)	(0 0 0 1 1 1)

$v_5 = u_2$   
 $v_4 = u_1$   
 $v_3 = u_0$

$u_6 \ u_1 \ u_2 \quad v_6 \ v_1 \ v_2 \ v_3 \ v_4 \ v_5$

and what is  $v_2$ ;  $v_2$  was  $u_0$  plus  $u_1$ . Now let's



(Refer Slide Time 08:22)

**Block Codes**

- Example: Let  $k = 3$  and  $n = 6$ . The following table gives a block code of length 6. The code rate is  $R = \frac{1}{2}$ .

Message	Codewords
(0 0 0)	(0 0 0 0 0 0)
(1 0 0)	(0 1 1 1 0 0)
(0 1 0)	(1 0 1 0 1 0)
(1 1 0)	(1 1 0 1 1 0)
(0 0 1)	(1 1 0 0 0 1)
(1 0 1)	(1 0 1 1 0 1)
(0 1 1)	(0 1 1 0 1 1)
(1 1 1)	(0 0 0 1 1 1)

$v_5 = u_2$   
 $v_4 = u_1$   
 $v_3 = u_0$   
 $v_2 = u_0 + u_1$

$u_0 \ u_1 \ u_2$      $v_0 \ v_1 \ v_2 \ v_3 \ v_4 \ v_5$

look at  $v_1$ . If we look at these two,  $u_0$  plus  $u_2$ ,

(Refer Slide Time 09:03)

**Block Codes**

- Example: Let  $k = 3$  and  $n = 6$ . The following table gives a block code of length 6. The code rate is  $R = \frac{1}{2}$ .

Message	Codewords
(0 0 0)	(0 0 0 0 0 0)
(1 0 0)	(0 1 1 1 0 0)
(0 1 0)	(1 0 1 0 1 0)
(1 1 0)	(1 1 0 1 1 0)
(0 0 1)	(1 1 0 0 0 1)
(1 0 1)	(1 0 1 1 0 1)
(0 1 1)	(0 1 1 0 1 1)
(1 1 1)	(0 0 0 1 1 1)

$v_5 = u_2$   
 $v_4 = u_1$   
 $v_3 = u_0$   
 $v_2 = u_0 + u_1$   
 $v_1 = u_0 + u_2$

$u_0 \ u_1 \ u_2$      $v_0 \ v_1 \ v_2 \ v_3 \ v_4 \ v_5$

so  $0$  plus  $0$  is  $0$ ,  $1$  plus  $0$  is  $1$ ,  $0$  plus  $0$  is  $0$ ,  $1$  plus  $0$  is  $1$ ,  $0$  plus  $1$  is  $1$ ,  $1$  plus  $1$  is  $0$ ,  $0$  plus  $1$  is  $1$  and  $1$  plus  $1$  is  $0$ ; so  $v_1$  is nothing but  $u_0$  plus  $u_2$ , Ok. Now look at last this one  $v_0$  what is  $v_0$ , we can see that this is same as  $u_1$  plus  $u_2$ .

(Refer Slide Time 09:15)

**Block Codes**

- Example: Let  $k = 3$  and  $n = 6$ . The following table gives a block code of length 6. The code rate is  $R = \frac{1}{2}$ .

Message	Codewords
(0 0 0)	(0 0 0 0 0 0)
(1 0 0)	(0 1 1 1 0 0)
(0 1 0)	(1 0 1 0 1 0)
(1 1 0)	(1 1 0 1 1 0)
(0 0 1)	(1 1 0 0 0 1)
(1 0 1)	(1 0 1 1 0 1)
(0 1 1)	(0 1 1 0 1 1)
(1 1 1)	(0 0 0 1 1 1)

$v_5 = v_2$   
 $v_4 = v_1$   
 $v_3 = v_0$   
 $v_2 = u_0 + u_1$   
 $v_1 = u_0 + u_2$   
 $v_0 = u_1 + u_2$

$u_0 \ u_1 \ u_2$       $v_0 \ v_1 \ v_2 \ v_3 \ v_4 \ v_5$

This is  $u_1$  plus  $u_2$ . So 0 plus 0 is 0, 0 plus 0 is 0, 1 plus 0 is 1, 1 plus 0 is 1, 0 plus 1 is 1, 0 plus 1 is 1, 1 plus 1 is 0 and 1 plus 1 is 0. So this is how we have mapped our information

(Refer Slide Time 09:38)

**Block Codes**

- Example: Let  $k = 3$  and  $n = 6$ . The following table gives a block code of length 6. The code rate is  $R = \frac{1}{2}$ .

Message	Codewords
(0 0 0)	(0 0 0 0 0 0)
(1 0 0)	(0 1 1 1 0 0)
(0 1 0)	(1 0 1 0 1 0)
(1 1 0)	(1 1 0 1 1 0)
(0 0 1)	(1 1 0 0 0 1)
(1 0 1)	(1 0 1 1 0 1)
(0 1 1)	(0 1 1 0 1 1)
(1 1 1)	(0 0 0 1 1 1)

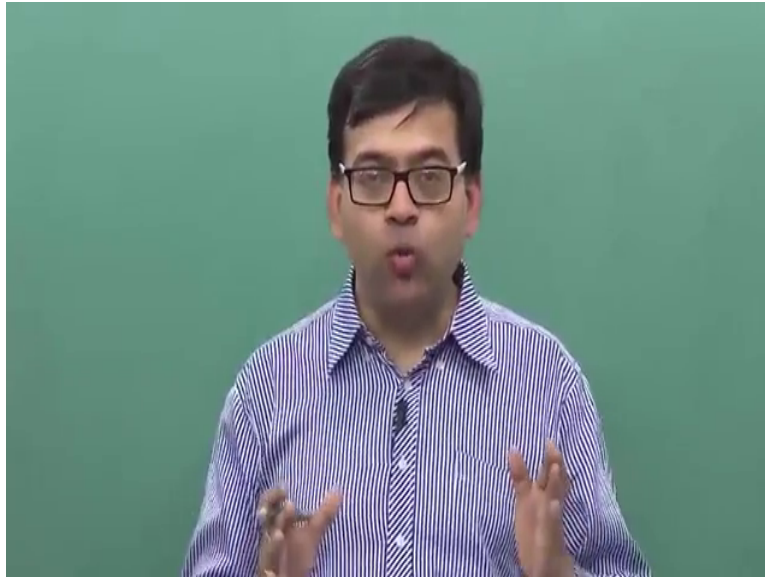
$v_5 = v_2$   
 $v_4 = v_1$   
 $v_3 = v_0$   
 $v_2 = u_0 + u_1$   
 $v_1 = u_0 + u_2$   
 $v_0 = u_1 + u_2$

$u_0 \ u_1 \ u_2$       $v_0 \ v_1 \ v_2 \ v_3 \ v_4 \ v_5$

bits into our coded bits

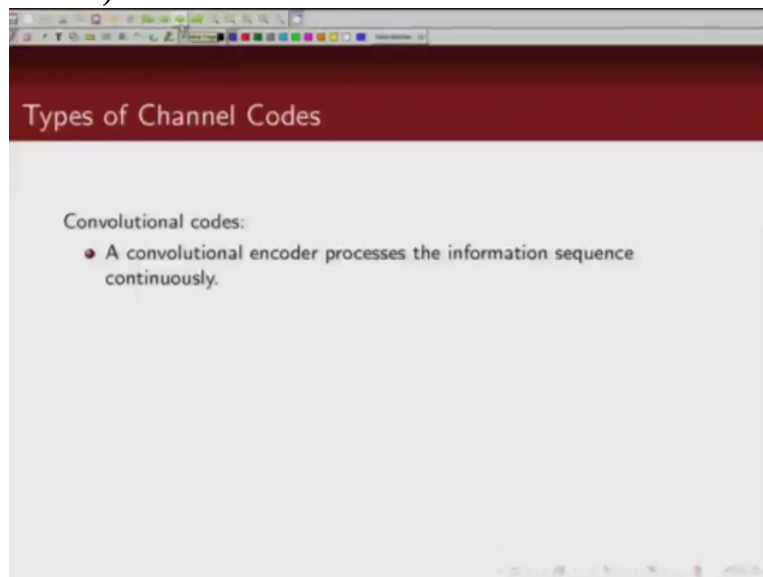
Ok, so again to recap, in block codes we take,

(Refer Slide Time 09:49)



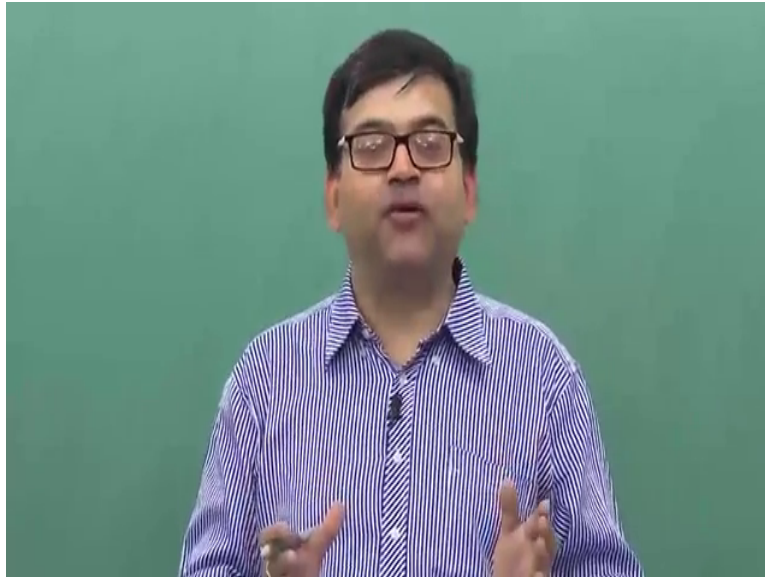
we partition our information sequence into blocks of  $k$ -bits and we map these  $k$ -bits into blocks of  $n$ -bits and this mapping is memoryless. In other words, how we map these  $k$ -bits does not depend upon how we have mapped the previous blocks of  $k$ -bits. Ok

(Refer Slide Time 10:16)



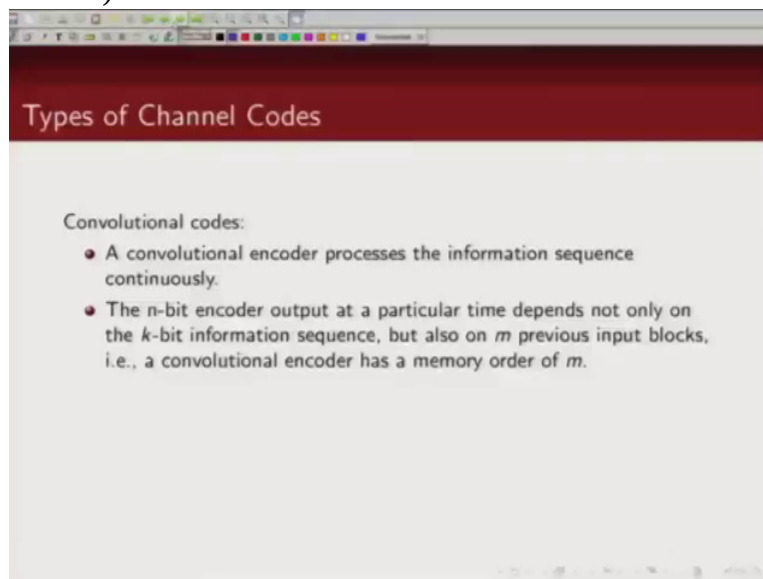
so let's now contrast it with what are convolutional codes and how are they different from convolutional codes. So in block codes we parse our information sequence

(Refer Slide Time 10:28)



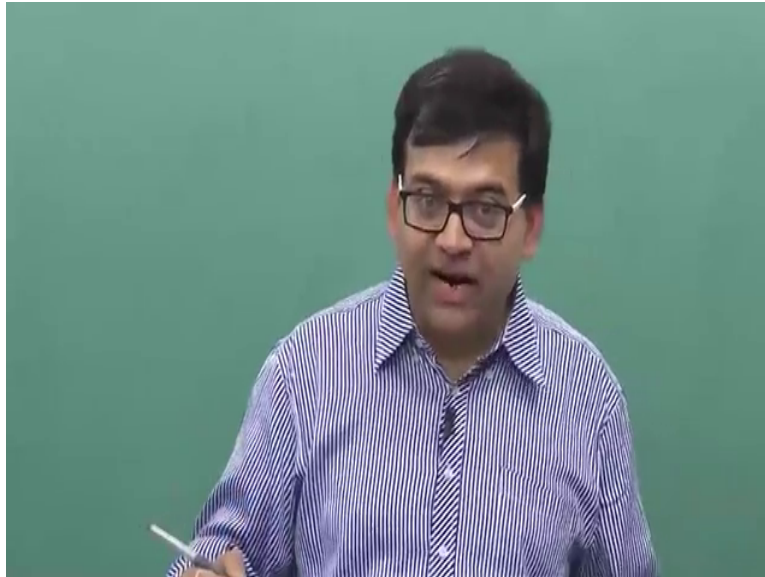
into blocks of data and we handle them block by block, whereas in convolutional code, you can process information sequence in a continuous fashion. The second difference

(Refer Slide Time 10:41)



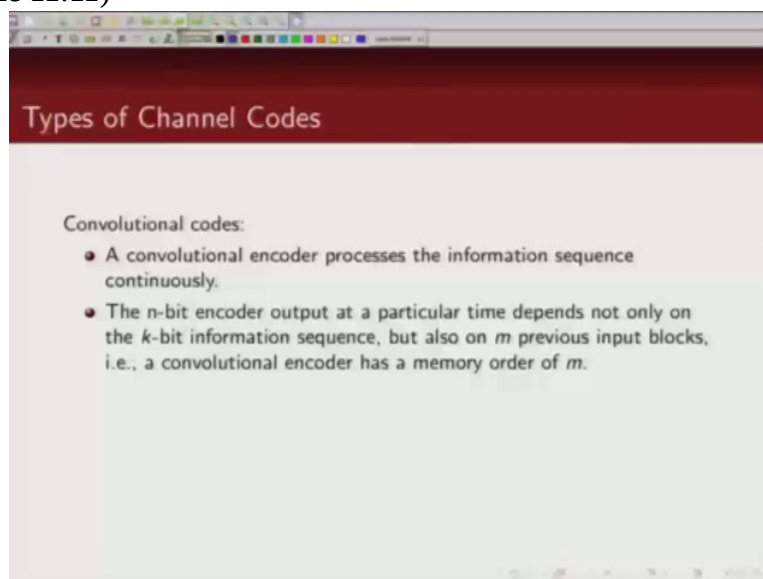
is the encoding in convolutional code is with memory. In other words the current output

(Refer Slide Time 10:51)



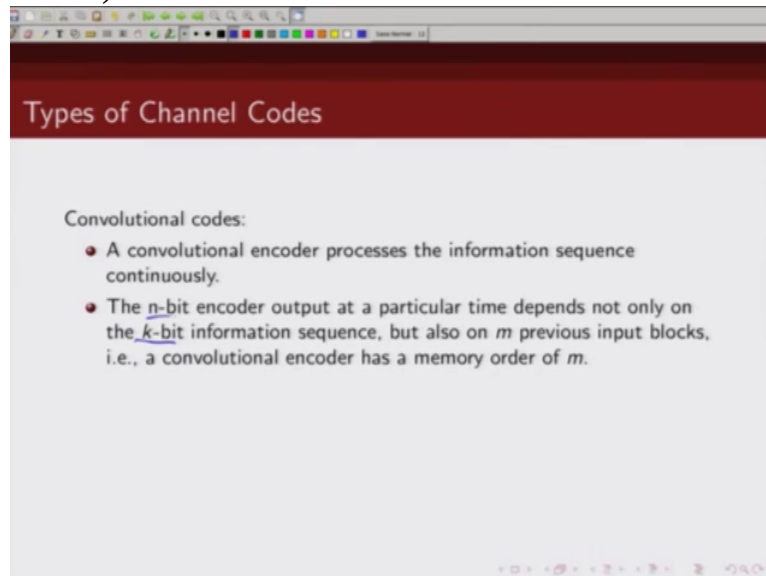
not only depends on current input but it also depends on past inputs and outputs, Ok. So unlike block codes, in convolutional codes, output depends on past inputs and outputs. So

(Refer Slide Time 11:11)



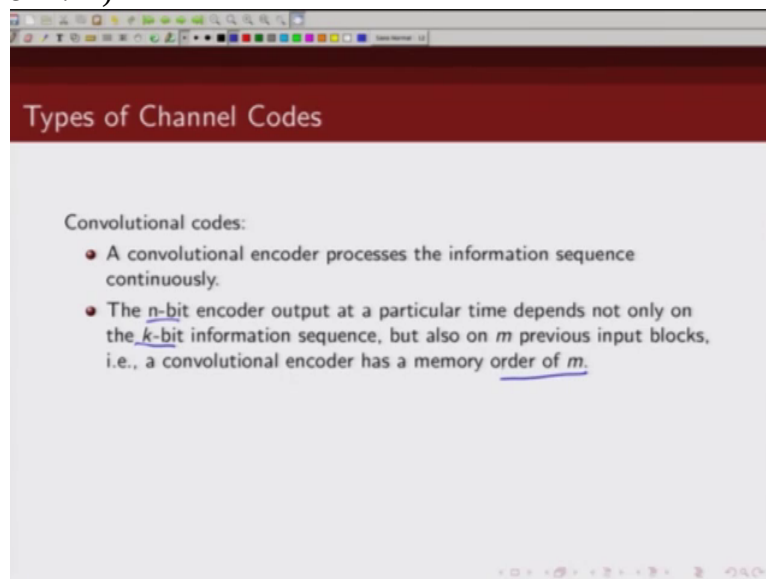
if we have  $n \times k$  convolutional codes where  $k$  is number of information bits,

(Refer Slide Time 11:18)



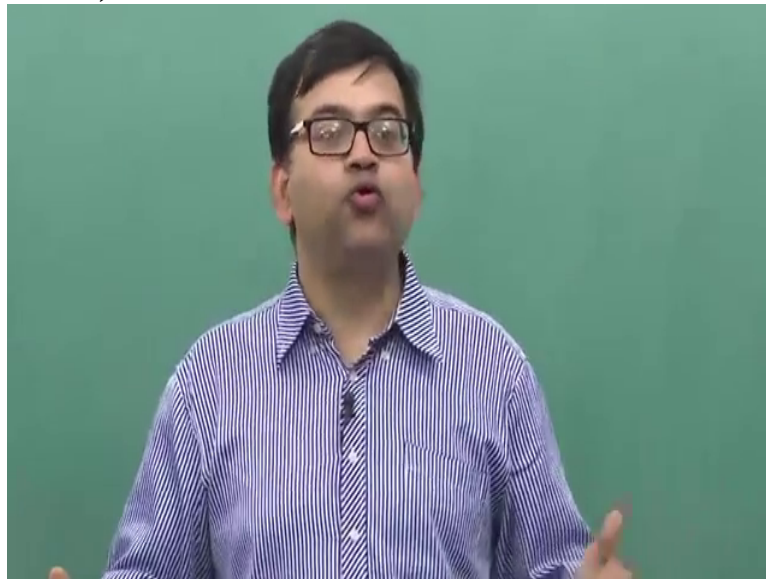
$n$  is the number of coded bits we have another parameter,

(Refer Slide Time 11:24)



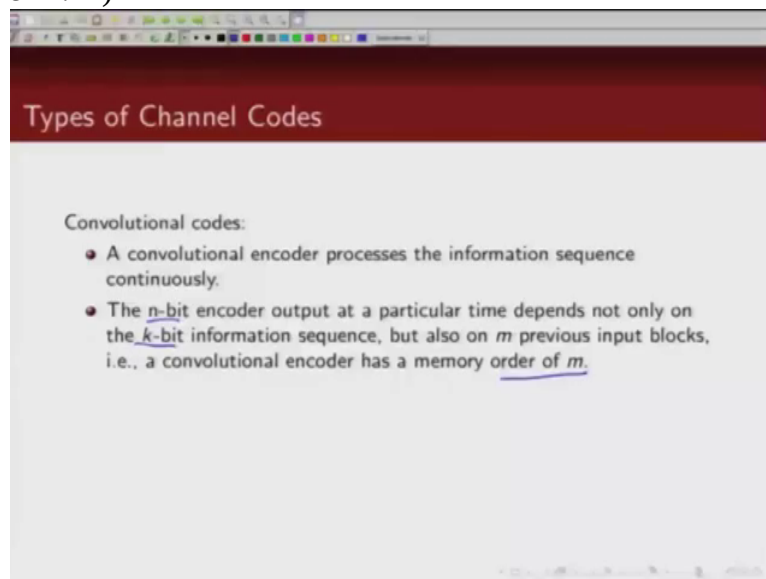
we are calling it memory order which signifies basically how many past bits or how many, what's past information that has been used

(Refer Slide Time 11:37)



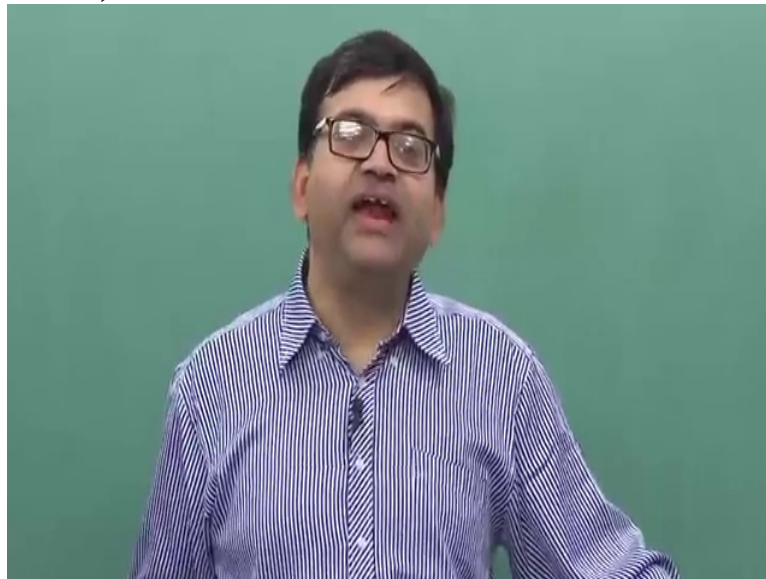
to generate the current output

(Refer Slide Time 11:42)



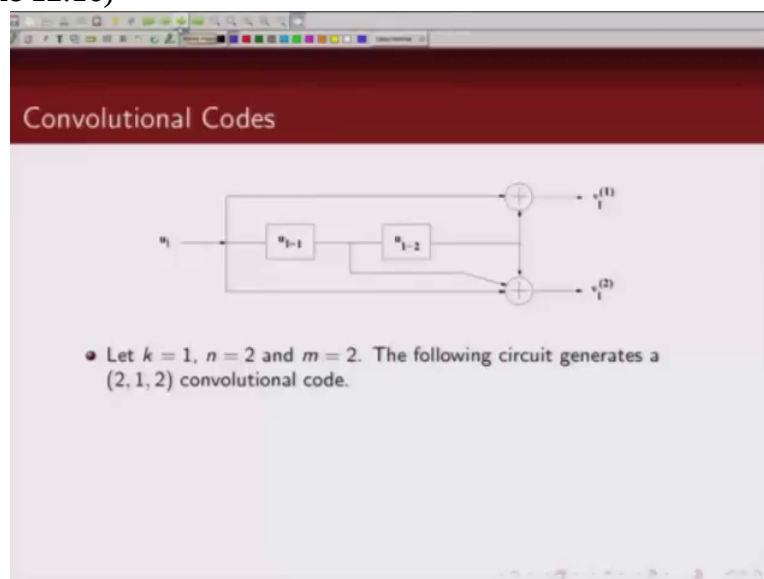
So we define convolutional code not only by these parameters  $n$  and  $k$  but another parameter which basically denotes the memory of the encoder. Another subtle difference,

(Refer Slide Time 12:03)



in case of convolutional codes typically the values of  $k$  and  $n$  are much smaller compared to values of  $k$  and  $n$  for block codes.

(Refer Slide Time 12:16)



So let's take an example now for convolutional code. So here we have one input



(Refer Slide Time 12:22)

Convolutional Codes

The diagram shows an input  $u_1$  entering a circuit. The input splits into two paths. The upper path goes through a block labeled  $u_{1-1}$ , then through a block labeled  $u_{1-2}$ , and finally to an adder  $\oplus$ . The lower path goes through a block labeled  $u_{1-1}$  and then to another adder  $\oplus$ . The output of the upper adder is  $v_1^{(1)}$  and the output of the lower adder is  $v_1^{(2)}$ .

- Let  $k = 1$ ,  $n = 2$  and  $m = 2$ . The following circuit generates a  $(2, 1, 2)$  convolutional code.

and two outputs.

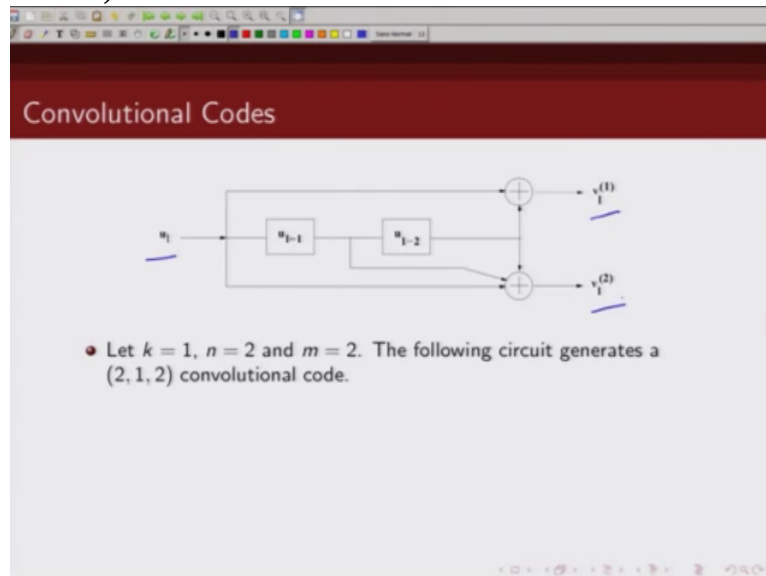
(Refer Slide Time 12:25)

Convolutional Codes

The diagram shows an input  $u_1$  entering a circuit. The input splits into two paths. The upper path goes through a block labeled  $u_{1-1}$ , then through a block labeled  $u_{1-2}$ , and finally to an adder  $\oplus$ . The lower path goes through a block labeled  $u_{1-1}$  and then to another adder  $\oplus$ . The output of the upper adder is  $v_1^{(1)}$  and the output of the lower adder is  $v_1^{(2)}$ .

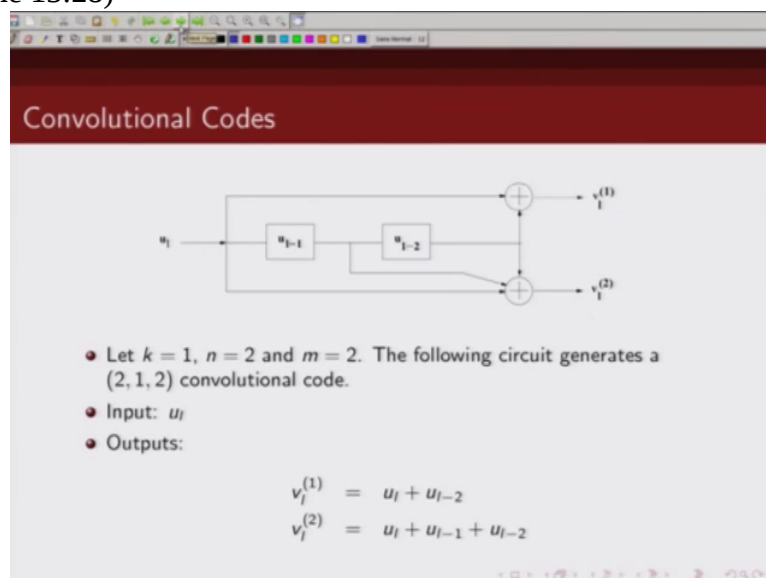
- Let  $k = 1$ ,  $n = 2$  and  $m = 2$ . The following circuit generates a  $(2, 1, 2)$  convolutional code.

(Refer Slide Time 12:26)



The input we are denoting by  $u$  of  $l$ , output we are denoting by  $v$  of  $l$  1,  $v$  of  $l$  2. Now note here, each of the outputs here not only depends on the current input which is  $u$  1 but it also depends on these past values. It also depends on what  $u$  1 minus 1 was, what  $u$  1 minus 2 was; so this an example of memory order 2. So the current input, current output not only depends on current input but also depends on past two values of the input. So this is an example of a 2 1 2 convolutional code.  $n$  is 2, there are two outputs,  $k$  is 1, one input and memory order is 2 because the output depends on past two values of information sequence. So you can see here,

(Refer Slide Time 13:28)



the first input which is  $v$  1,  $v$  1 1, it is basically  $u$  1 plus  $u$  1 minus 2. So in other words, it depends on the current input and what was the input past two values basically and similarly

this one depends on current input, past input, one past input and the, this  $u_{l-1}$  minus 2. So this is basically how, so you can see

(Refer Slide Time 14:03)

**Convolutional Codes**

The diagram shows an input  $u_l$  entering a circuit with two delay elements,  $u_{l-1}$  and  $u_{l-2}$ . The input  $u_l$  is added to  $u_{l-2}$  at a summing junction to produce output  $v_l^{(1)}$ . Simultaneously,  $u_l$  is added to both  $u_{l-1}$  and  $u_{l-2}$  at another summing junction to produce output  $v_l^{(2)}$ .

- Let  $k = 1$ ,  $n = 2$  and  $m = 2$ . The following circuit generates a  $(2, 1, 2)$  convolutional code.
- Input:  $u_l$
- Outputs:

$$\begin{aligned} v_l^{(1)} &= u_l + u_{l-2} \\ v_l^{(2)} &= u_l + u_{l-1} + u_{l-2} \end{aligned}$$

the difference here The output not only depends on current input but it also depends on past inputs. Similarly here, basically you can see the, in the convolutional code the output depends on past inputs and outputs, Ok. So that is one of the major difference between convolutional codes and block codes.

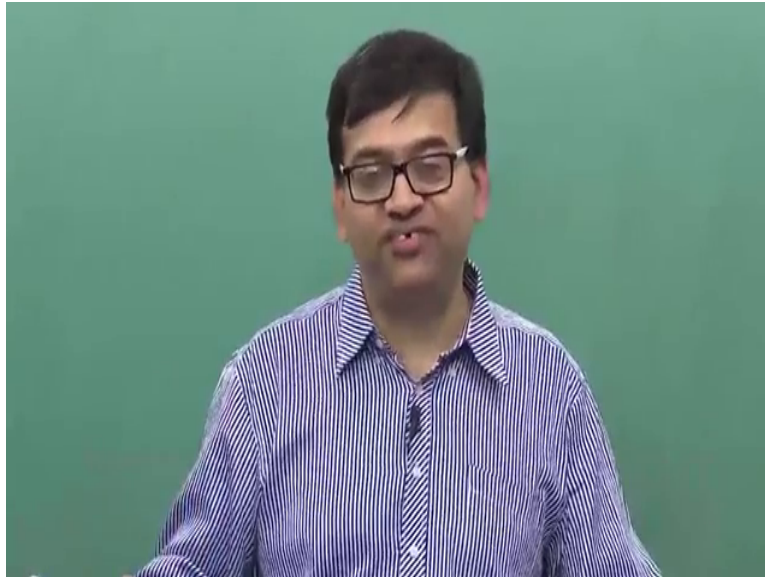
(Refer Slide Time 14:34)

**Decoding Strategies**

- The decoder produces an estimate  $\hat{u}$  of the information sequence based on the received sequence  $r$ .

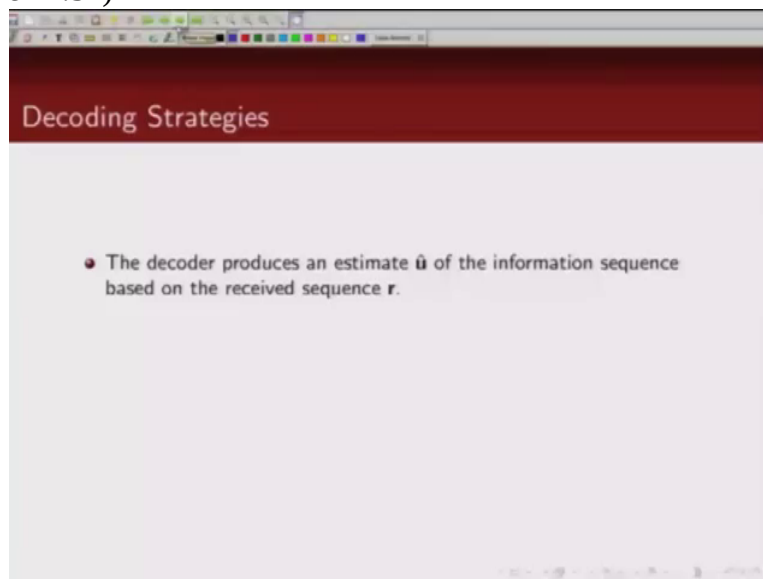
Now let's move to the topic of what sort of decoding strategy

(Refer Slide Time 14:42)



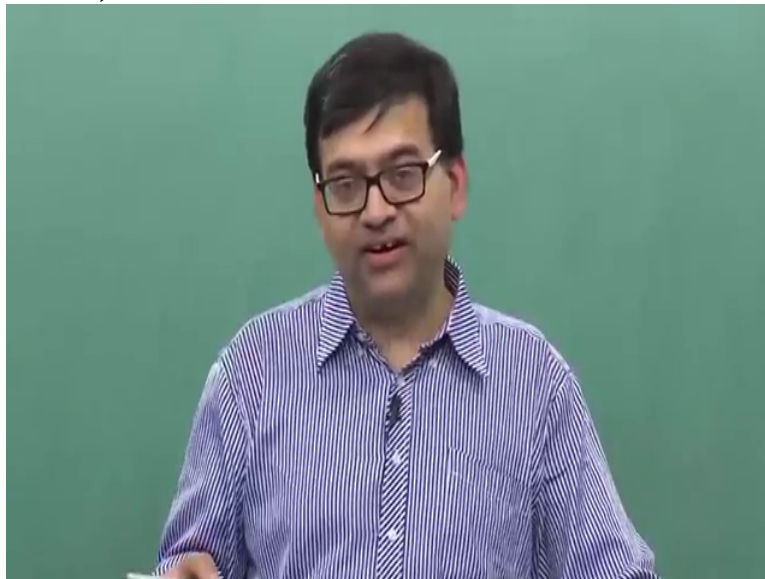
should we employ when we want to decode a code. So

(Refer Slide Time 14:51)



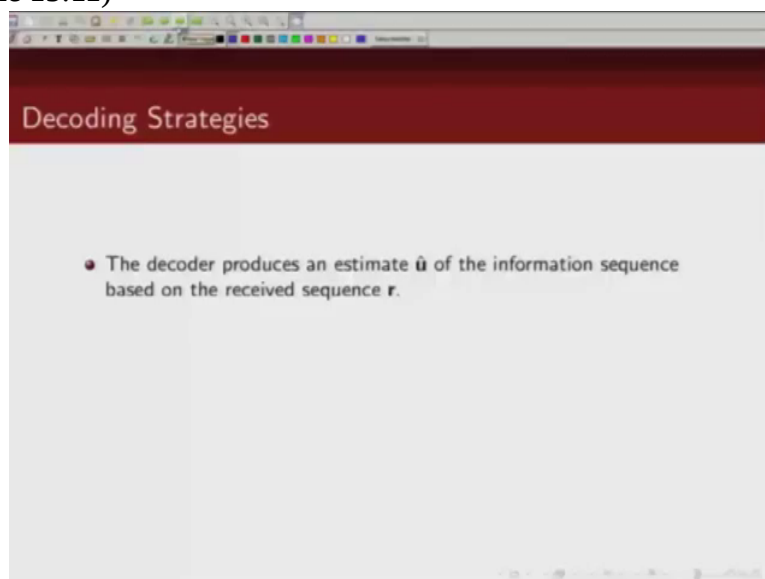
now, as I said, a decoder objective is it takes as input the demodulated signal  $r$

(Refer Slide Time 15:00)



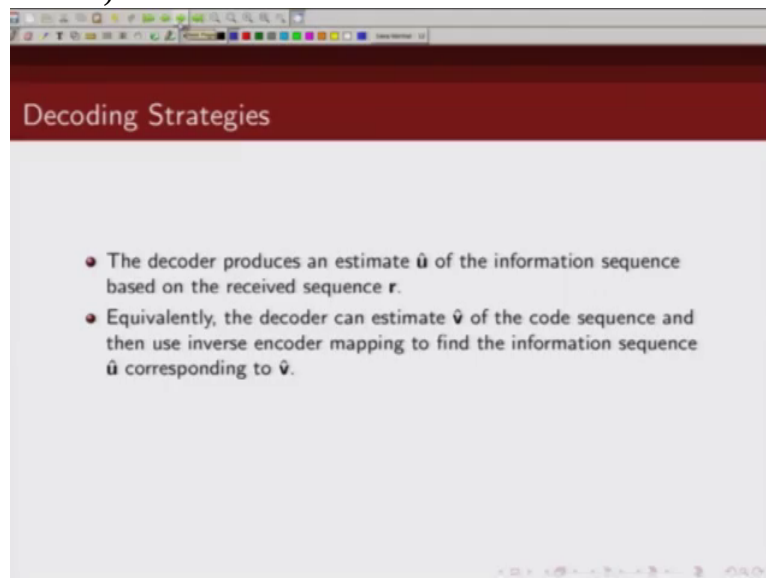
and it has to produce an estimate of the information sequence you had, right? So the decoder produces an estimate

(Refer Slide Time 15:11)



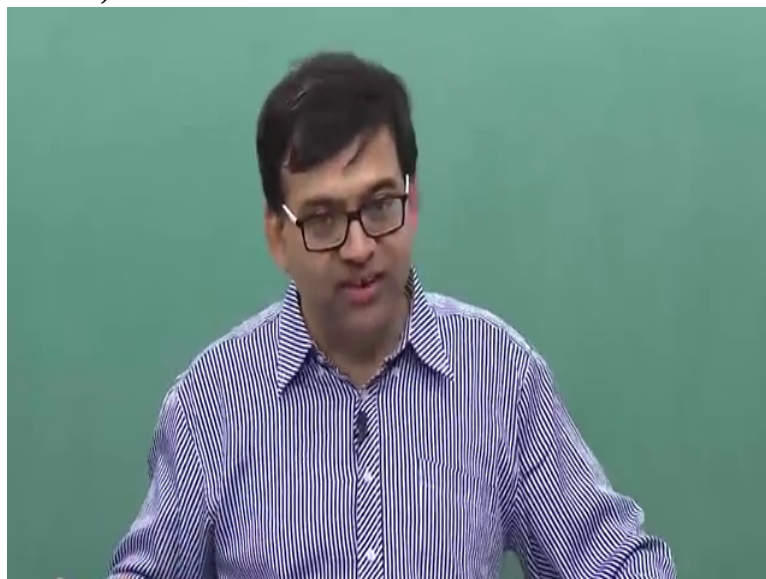
of the information sequence based on what it has received of demodulated output which is  $r$ .  
Now

(Refer Slide Time 15:21)



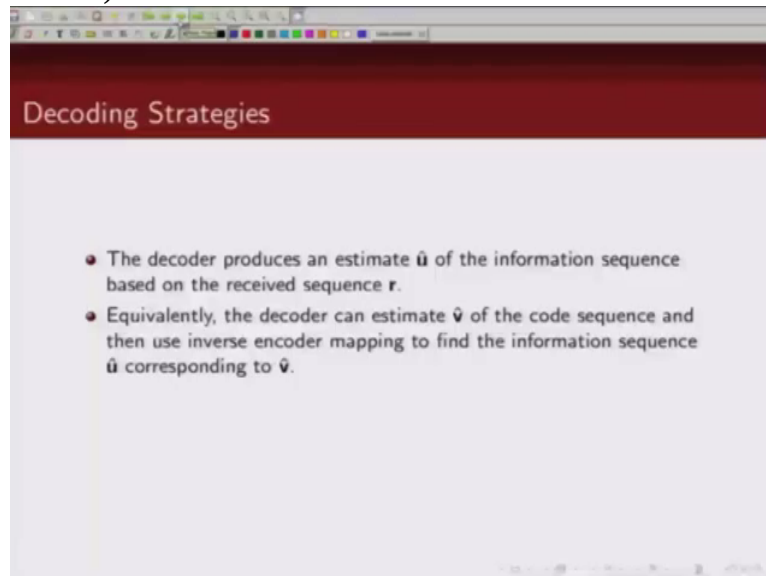
we can see this estimation of the information sequence problem is equivalent to estimating the code sequence

(Refer Slide Time 15:30)



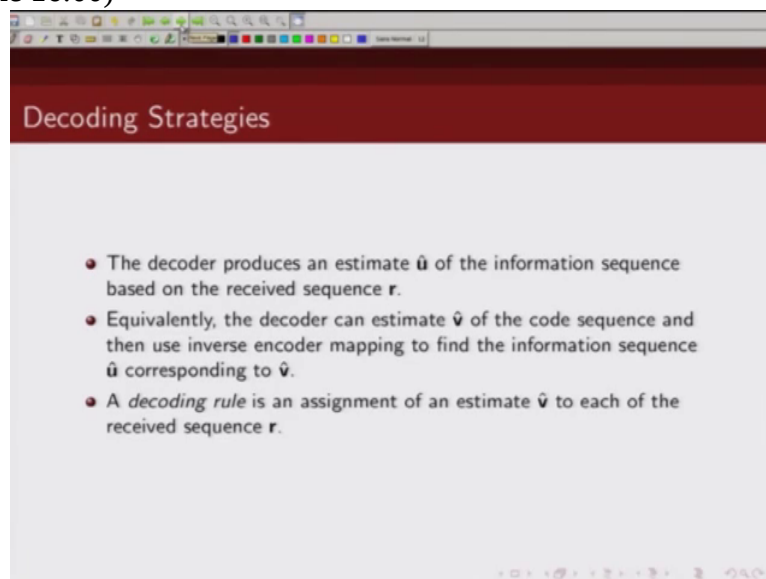
because there is one to one mapping from a particular codeword to the information sequence  
So we can say equivalently the problem that

(Refer Slide Time 15:45)



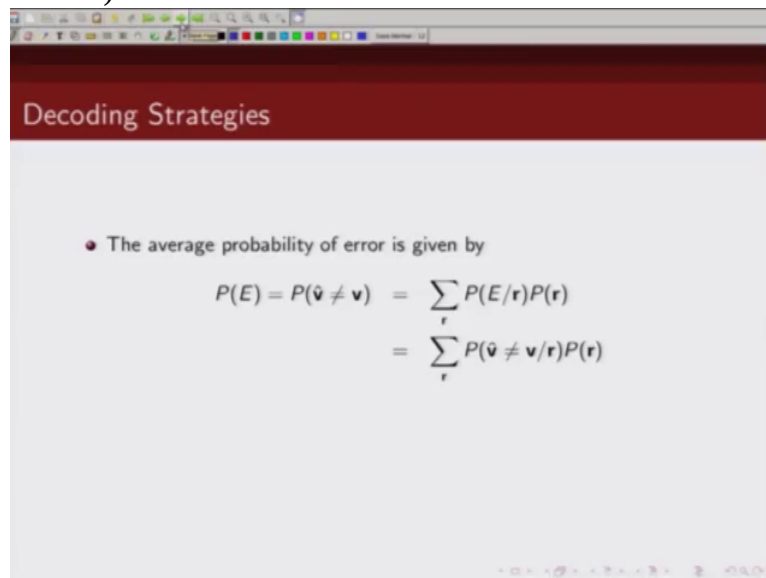
decoder has to estimate is it has to estimate the code sequence, given a received sequence  $\mathbf{r}$  because there is one to one mapping from the message, message bits to the code bits; so what is a decoding

(Refer Slide Time 16:00)



strategy or what is a decoding rule? A decoding rule is nothing but given a received sequence  $\mathbf{r}$  we are trying to estimate what our code sequence, transmitted code sequence was. So we are trying to estimate  $\hat{\mathbf{v}}$  or  $\hat{\mathbf{u}}$  from received sequence  $\mathbf{r}$ . So we have to decide how, what rule or what logic should we use when we get received sequence  $\mathbf{r}$ , how do we assign that received sequence  $\mathbf{r}$  to any particular codeword. Now

(Refer Slide Time 16:38)



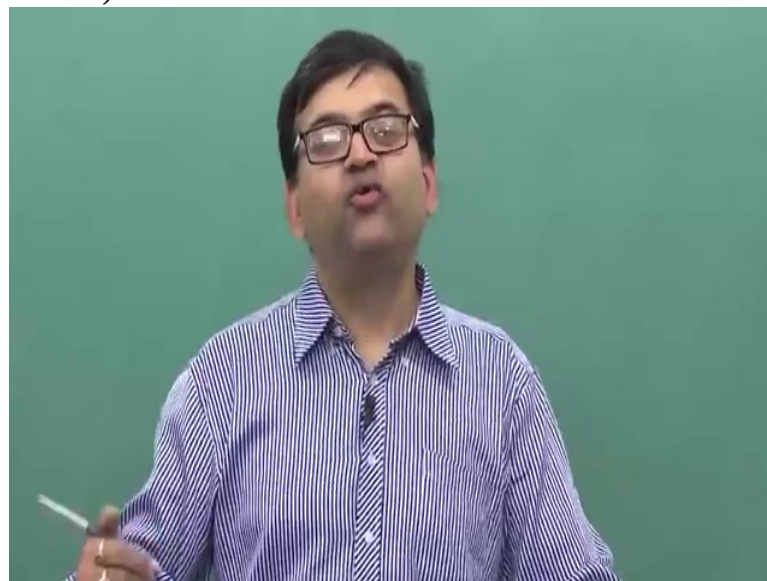
Decoding Strategies

- The average probability of error is given by

$$P(E) = P(\hat{\mathbf{v}} \neq \mathbf{v}) = \sum_r P(E/r)P(r)$$
$$= \sum_r P(\hat{\mathbf{v}} \neq \mathbf{v}/r)P(r)$$

one of the policies which we can use is basically

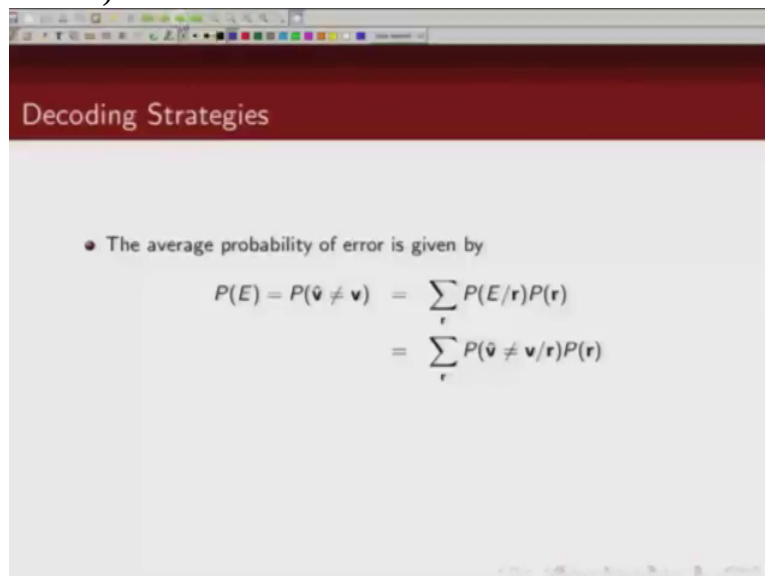
(Refer Slide Time 16:41)



to minimize probability of error Now when does an error occur; when my decoded sequence is not same as my transmitted signal; so my probability



(Refer Slide Time 16:53)



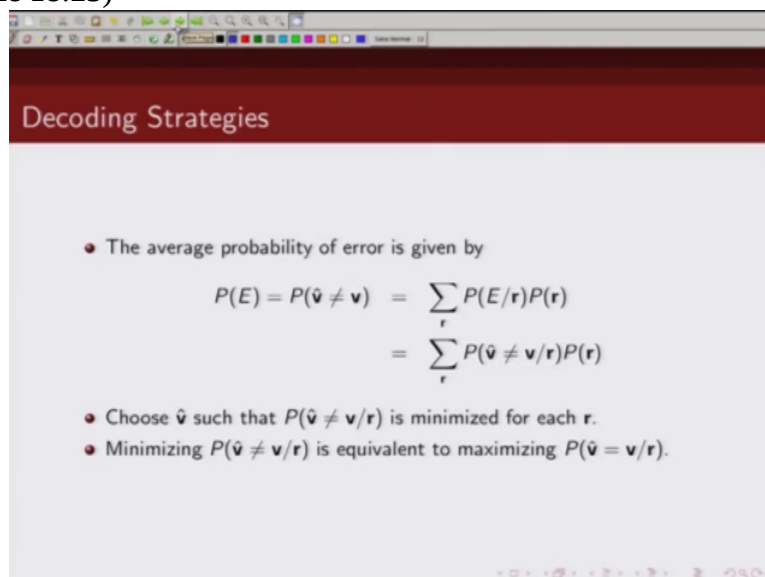
Decoding Strategies

- The average probability of error is given by

$$P(E) = P(\hat{\mathbf{v}} \neq \mathbf{v}) = \sum_{\mathbf{r}} P(E/\mathbf{r})P(\mathbf{r})$$
$$= \sum_{\mathbf{r}} P(\hat{\mathbf{v}} \neq \mathbf{v}/\mathbf{r})P(\mathbf{r})$$

of error is given by probability then, when my estimated sequence which I denote by  $\hat{\mathbf{v}}$  is not same as  $\mathbf{v}$ . So this can be written as probability of error given  $\mathbf{r}$  received sequence multiplied by probability of the received sequence  $\mathbf{r}$  and sum over all possible received sequence. And error is nothing but, when  $\mathbf{v}$  is not same as  $\hat{\mathbf{v}}$ , so I can write this equation in this particular form. So if I want to minimize probability of error, I will have to minimize this. So my decoding rule should be such that this is minimized. So there are two terms in this. One is  $P$  of  $\mathbf{r}$  and another is this term. Now whatever  $\hat{\mathbf{v}}$  I choose, that does not change  $P$  of  $\mathbf{r}$ . So the choice of decoding rule does not change my  $P$  of  $\mathbf{r}$ . So in other words if I have to minimize probability of error I should choose my  $\hat{\mathbf{v}}$  in such a way such that this is minimized. For each received sequence  $\mathbf{r}$ , this term should be minimized, Ok? Now

(Refer Slide Time 18:23)



Decoding Strategies

- The average probability of error is given by

$$P(E) = P(\hat{\mathbf{v}} \neq \mathbf{v}) = \sum_{\mathbf{r}} P(E/\mathbf{r})P(\mathbf{r})$$
$$= \sum_{\mathbf{r}} P(\hat{\mathbf{v}} \neq \mathbf{v}/\mathbf{r})P(\mathbf{r})$$

- Choose  $\hat{\mathbf{v}}$  such that  $P(\hat{\mathbf{v}} \neq \mathbf{v}/\mathbf{r})$  is minimized for each  $\mathbf{r}$ .
- Minimizing  $P(\hat{\mathbf{v}} \neq \mathbf{v}/\mathbf{r})$  is equivalent to maximizing  $P(\hat{\mathbf{v}} = \mathbf{v}/\mathbf{r})$ .

minimizing this term, minimizing this term is same as maximizing this term,

(Refer Slide Time 18:36)

Decoding Strategies

- The average probability of error is given by
$$P(E) = P(\hat{\mathbf{v}} \neq \mathbf{v}) = \sum_{\mathbf{r}} P(E/\mathbf{r})P(\mathbf{r})$$
$$= \sum_{\mathbf{r}} \underline{P(\hat{\mathbf{v}} \neq \mathbf{v}/\mathbf{r})P(\mathbf{r})}$$
- Choose  $\hat{\mathbf{v}}$  such that  $P(\hat{\mathbf{v}} \neq \mathbf{v}/\mathbf{r})$  is minimized for each  $\mathbf{r}$ .
- Minimizing  $P(\hat{\mathbf{v}} \neq \mathbf{v}/\mathbf{r})$  is equivalent to maximizing  $P(\hat{\mathbf{v}} = \mathbf{v}/\mathbf{r})$ .

correct? Minimizing the probability  $\hat{v}$  is not same as  $v$  given  $r$  is equivalent to maximizing the probability that  $\hat{v}$  is equal to  $v$  given  $r$  ok so

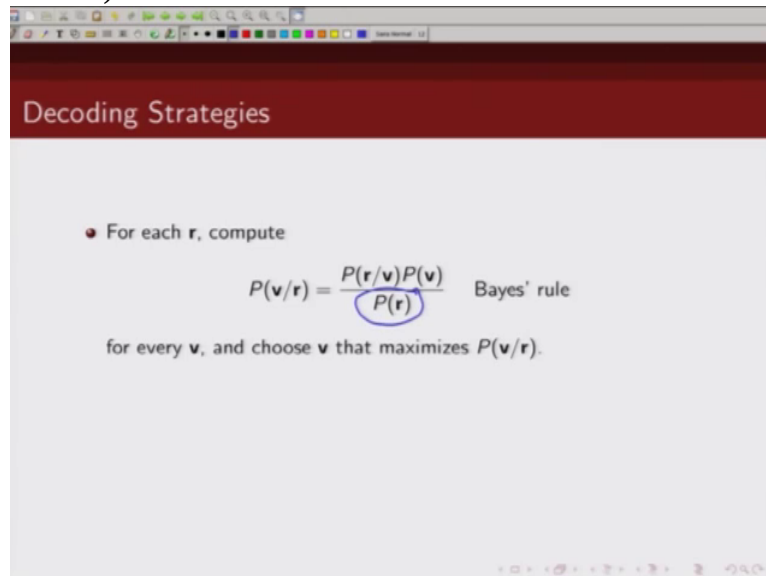
(Refer Slide Time 18:53)

Decoding Strategies

- For each  $\mathbf{r}$ , compute
$$P(\mathbf{v}/\mathbf{r}) = \frac{P(\mathbf{r}/\mathbf{v})P(\mathbf{v})}{P(\mathbf{r})} \quad \text{Bayes' rule}$$
- for every  $\mathbf{v}$ , and choose  $\mathbf{v}$  that maximizes  $P(\mathbf{v}/\mathbf{r})$ .

we have to maximize this. Now using Bayes rule we can write probability of  $v$  given  $r$  as probability of  $r$  given  $v$  multiplied by probability of  $v$  divided by probability of  $r$ . And this has to be maximized for every basically  $v$ , so we should choose our  $v$  such that this thing is maximized. Now again choice of  $v$  does not change this.

(Refer Slide Time 19:22)



Decoding Strategies

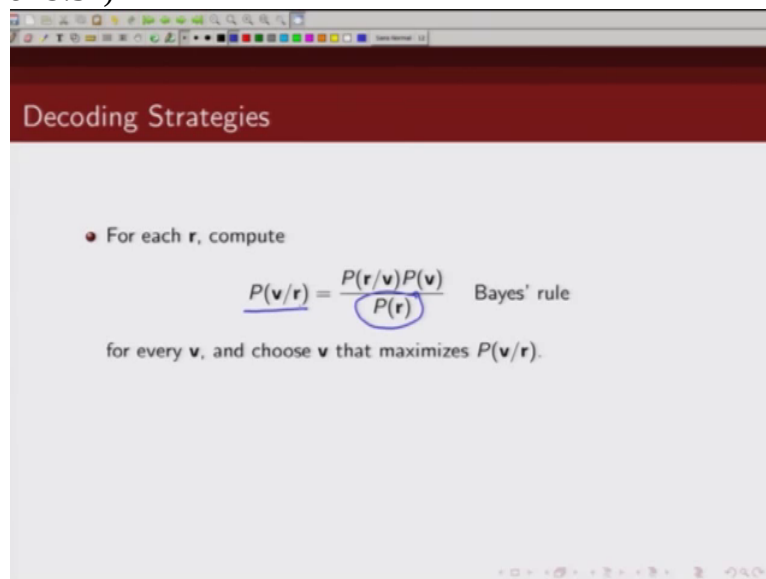
- For each  $r$ , compute

$$P(\mathbf{v}/r) = \frac{P(r/\mathbf{v})P(\mathbf{v})}{P(r)} \quad \text{Bayes' rule}$$

for every  $\mathbf{v}$ , and choose  $\mathbf{v}$  that maximizes  $P(\mathbf{v}/r)$ .

So we can write our probability to maximize,

(Refer Slide Time 19:32)



Decoding Strategies

- For each  $r$ , compute

$$P(\mathbf{v}/r) = \frac{P(r/\mathbf{v})P(\mathbf{v})}{P(r)} \quad \text{Bayes' rule}$$

for every  $\mathbf{v}$ , and choose  $\mathbf{v}$  that maximizes  $P(\mathbf{v}/r)$ .

so to maximize this then becomes maximizing this quantity.

(Refer Slide Time 19:38)

Decoding Strategies

- For each  $\mathbf{r}$ , compute

$$P(\mathbf{v}/\mathbf{r}) = \frac{P(\mathbf{r}/\mathbf{v})P(\mathbf{v})}{P(\mathbf{r})} \quad \text{Bayes' rule}$$

for every  $\mathbf{v}$ , and choose  $\mathbf{v}$  that maximizes  $P(\mathbf{v}/\mathbf{r})$ .

(Refer Slide Time 19:40)

Decoding Strategies

- For each  $\mathbf{r}$ , compute

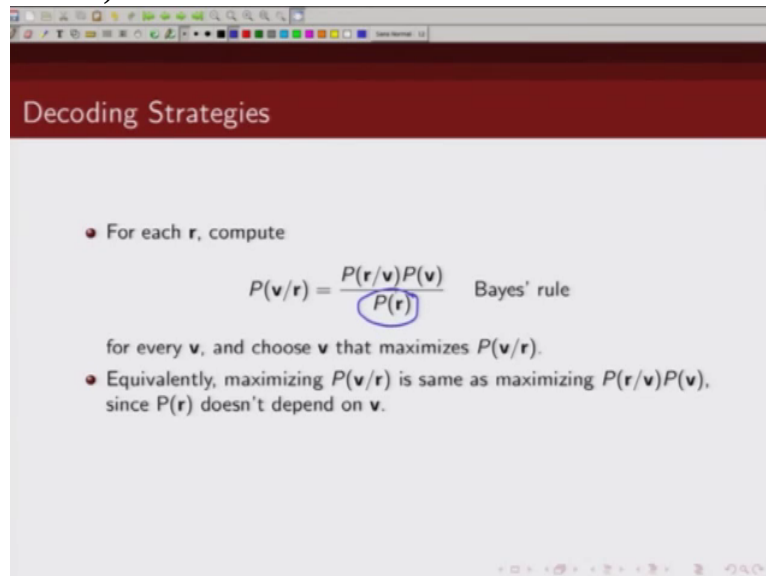
$$P(\mathbf{v}/\mathbf{r}) = \frac{P(\mathbf{r}/\mathbf{v})P(\mathbf{v})}{P(\mathbf{r})} \quad \text{Bayes' rule}$$

for every  $\mathbf{v}$ , and choose  $\mathbf{v}$  that maximizes  $P(\mathbf{v}/\mathbf{r})$ .

- Equivalently, maximizing  $P(\mathbf{v}/\mathbf{r})$  is same as maximizing  $P(\mathbf{r}/\mathbf{v})P(\mathbf{v})$ , since  $P(\mathbf{r})$  doesn't depend on  $\mathbf{v}$ .

So we can say maximizing this is nothing but maximizing this quantity because this quantity does not depend

(Refer Slide Time 19:51)



Decoding Strategies

- For each  $\mathbf{r}$ , compute

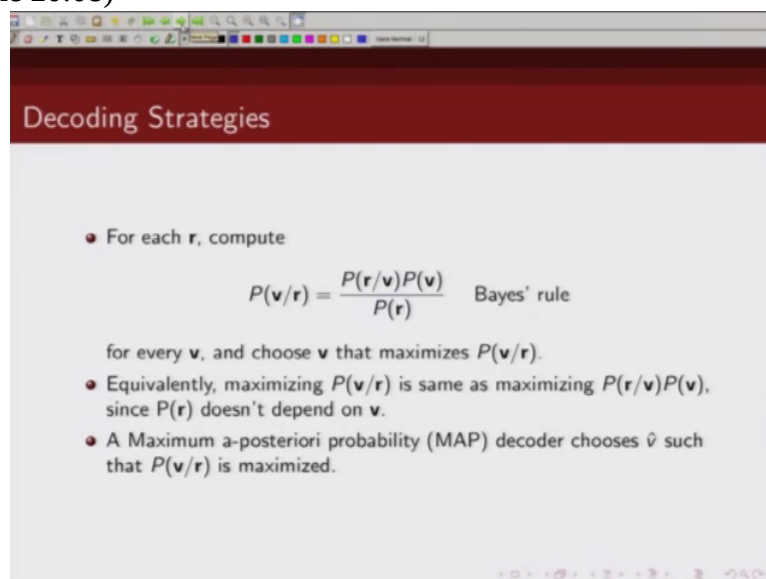
$$P(\mathbf{v}/\mathbf{r}) = \frac{P(\mathbf{r}/\mathbf{v})P(\mathbf{v})}{P(\mathbf{r})} \quad \text{Bayes' rule}$$

for every  $\mathbf{v}$ , and choose  $\mathbf{v}$  that maximizes  $P(\mathbf{v}/\mathbf{r})$ .

- Equivalently, maximizing  $P(\mathbf{v}/\mathbf{r})$  is same as maximizing  $P(\mathbf{r}/\mathbf{v})P(\mathbf{v})$ , since  $P(\mathbf{r})$  doesn't depend on  $\mathbf{v}$ .

on choice of  $\mathbf{v}$ , Ok So if you want to minimize probability of error you want to maximize this. We want to maximize this quantity.

(Refer Slide Time 20:08)



Decoding Strategies

- For each  $\mathbf{r}$ , compute

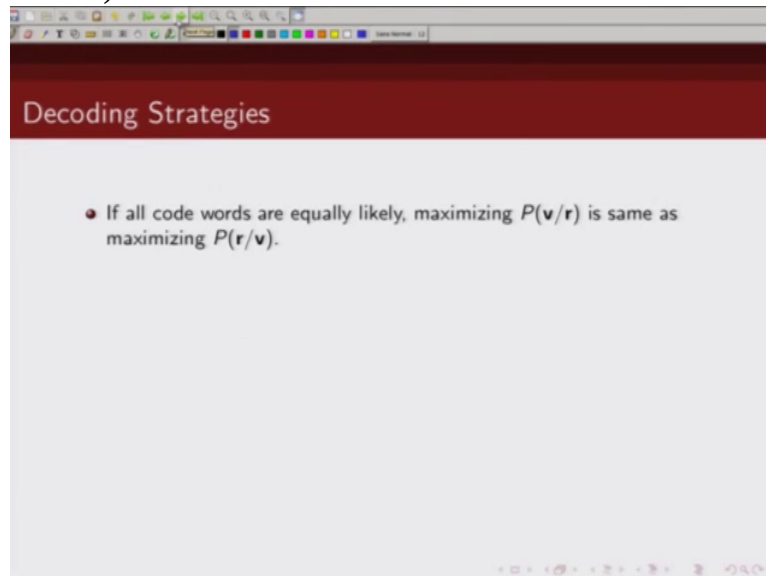
$$P(\mathbf{v}/\mathbf{r}) = \frac{P(\mathbf{r}/\mathbf{v})P(\mathbf{v})}{P(\mathbf{r})} \quad \text{Bayes' rule}$$

for every  $\mathbf{v}$ , and choose  $\mathbf{v}$  that maximizes  $P(\mathbf{v}/\mathbf{r})$ .

- Equivalently, maximizing  $P(\mathbf{v}/\mathbf{r})$  is same as maximizing  $P(\mathbf{r}/\mathbf{v})P(\mathbf{v})$ , since  $P(\mathbf{r})$  doesn't depend on  $\mathbf{v}$ .
- A Maximum a-posteriori probability (MAP) decoder chooses  $\hat{\mathbf{v}}$  such that  $P(\hat{\mathbf{v}}/\mathbf{r})$  is maximized.

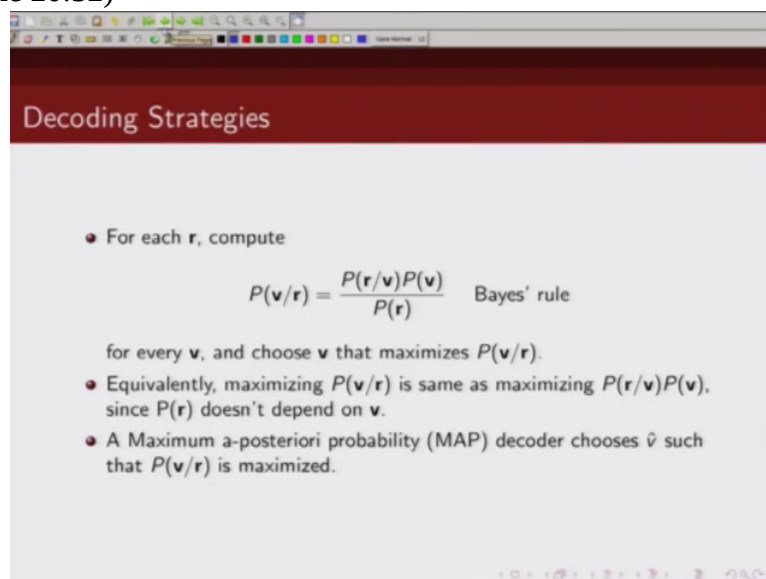
And MAP decoder, Maximum a posteriori probability decoder is the one which will do exactly that. It will choose a  $\hat{\mathbf{v}}$  such that this is, this probability is maximized. Now what

(Refer Slide Time 20:25)



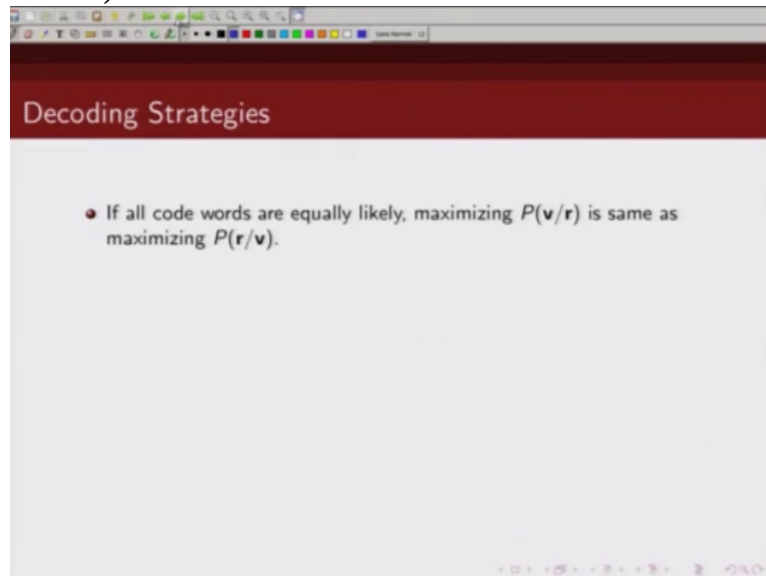
happens if all codewords are equally likely to happen? If all codewords are equally likely to happen, then look at

(Refer Slide Time 20:32)



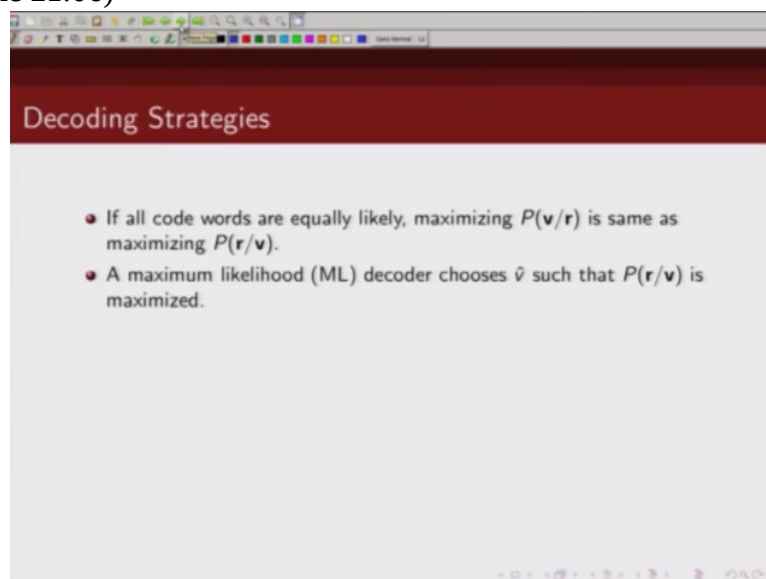
this term Probability of v will be same. So in that case maximizing probability of v given r is same as maximizing probability of r given v. So that's what we are saying.

(Refer Slide Time 20:49)



If all codewords are equally likely then maximizing probability of  $\mathbf{v}$  given  $\mathbf{r}$  is same as maximizing this likelihood ratio, likelihood function  $p$  of  $\mathbf{r}$  given  $\mathbf{v}$ ; so

(Refer Slide Time 21:06)



maximum likelihood decoder is the one which will choose  $\hat{\mathbf{v}}$  such that this quantity is maximized. Now

(Refer Slide Time 21:17)

Decoding Strategies

- If all code words are equally likely, maximizing  $P(\mathbf{v}/\mathbf{r})$  is same as maximizing  $P(\mathbf{r}/\mathbf{v})$ .
- A maximum likelihood (ML) decoder chooses  $\hat{v}$  such that  $P(\mathbf{r}/\mathbf{v})$  is maximized.
- For a discrete memoryless channel (DMC) where each received symbol depends only on the corresponding transmitted symbol

$$P(\mathbf{r}/\mathbf{v}) = \prod_i P(r_i/v_i)$$

if we consider that our channel is discrete memoryless channel, in other words we can write the probability for discrete memoryless channel we can write probability of  $v$  given  $r$  as product of each individual probabilities. If that happens, then we can further simplify our maximizing criteria. So we want to maximize this, is same as maximizing this. Now since  $\log$  of  $x$  is a monotonously increasing function of  $x$ , we can say maximizing this probability is same,

(Refer Slide Time 22:04)

Decoding Strategies

- If all code words are equally likely, maximizing  $P(\mathbf{v}/\mathbf{r})$  is same as maximizing  $P(\mathbf{r}/\mathbf{v})$ .
- A maximum likelihood (ML) decoder chooses  $\hat{v}$  such that  $P(\mathbf{r}/\mathbf{v})$  is maximized.
- For a discrete memoryless channel (DMC) where each received symbol depends only on the corresponding transmitted symbol

$$P(\mathbf{r}/\mathbf{v}) = \prod_i P(r_i/v_i)$$

- Since  $\log x$  is a monotone increasing function of  $x$ , maximizing  $P(\mathbf{r}/\mathbf{v})$  is equivalent to maximizing  $\log P(\mathbf{r}/\mathbf{v})$ .

is equivalent to maximizing  $\log$  of probability of  $r$  given  $v$  Now if we do that, then this product becomes summation, Ok? So then we can basically write this as, basically then  $\log$  of probability of  $r$  given  $v$  will become basically summation and this will be basically, of course this will be some  $\log$  term here,  $\log$  term here and this is basically



(Refer Slide Time 22:40)

**Decoding Strategies**

- If all code words are equally likely, maximizing  $P(\mathbf{v}/\mathbf{r})$  is same as maximizing  $P(\mathbf{r}/\mathbf{v})$ .
- A maximum likelihood (ML) decoder chooses  $\hat{\mathbf{v}}$  such that  $P(\mathbf{r}/\mathbf{v})$  is maximized.
- For a discrete memoryless channel (DMC) where each received symbol depends only on the corresponding transmitted symbol

$$P(\mathbf{r}/\mathbf{v}) = \prod_i P(r_i/v_i) \quad \log P(\mathbf{r}/\mathbf{v}) = \sum_i \log P(r_i/v_i)$$

- Since  $\log x$  is a monotone increasing function of  $x$ , maximizing  $P(\mathbf{r}/\mathbf{v})$  is equivalent to maximizing  $\log P(\mathbf{r}/\mathbf{v})$ .

much easier to compute, Ok

So let's take an example

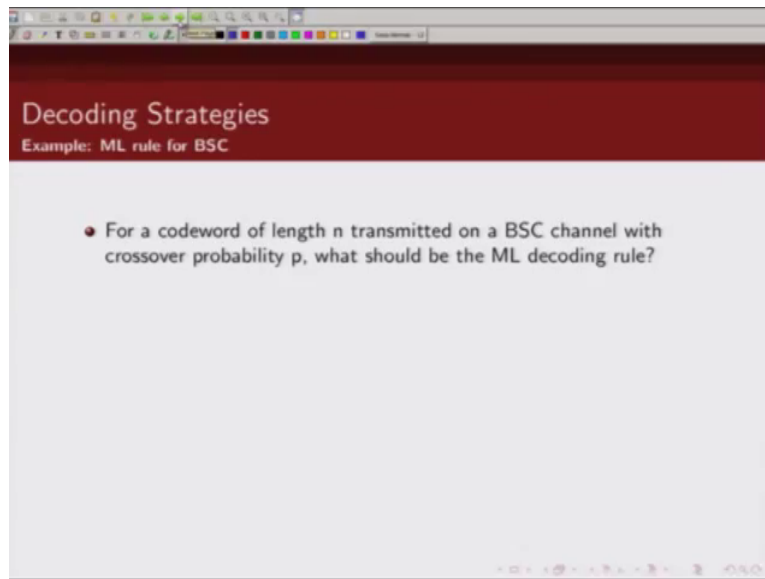
(Refer Slide Time 22:47)

**Decoding Strategies**  
Example: ML rule for BSC

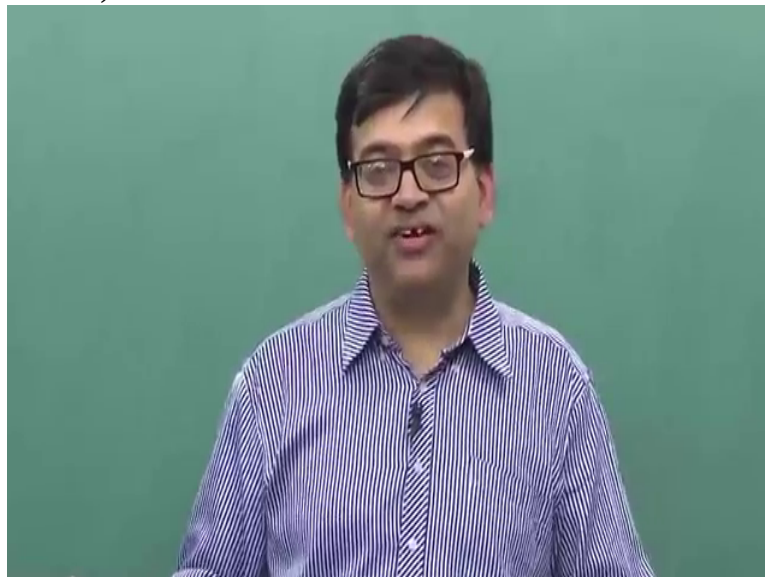
- For a codeword of length  $n$  transmitted on a BSC channel with crossover probability  $p$ , what should be the ML decoding rule?

We

(Refer Slide Time 22:50)

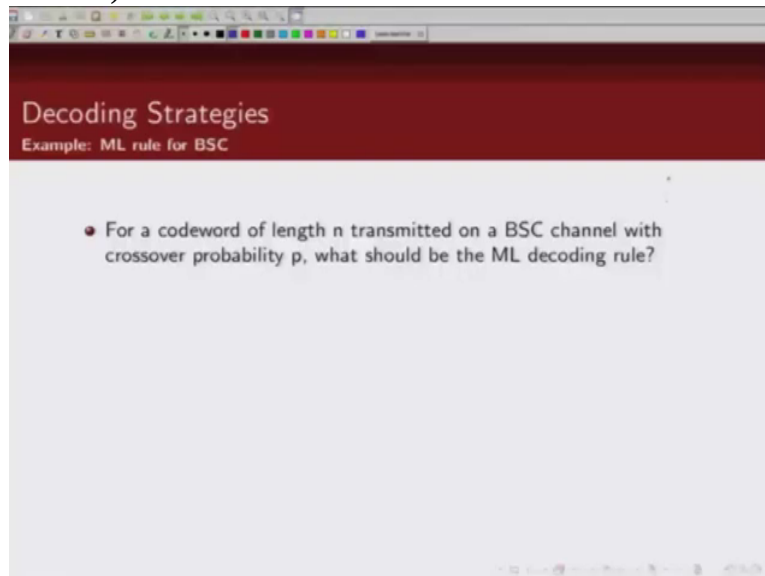


(Refer Slide Time 22:51)



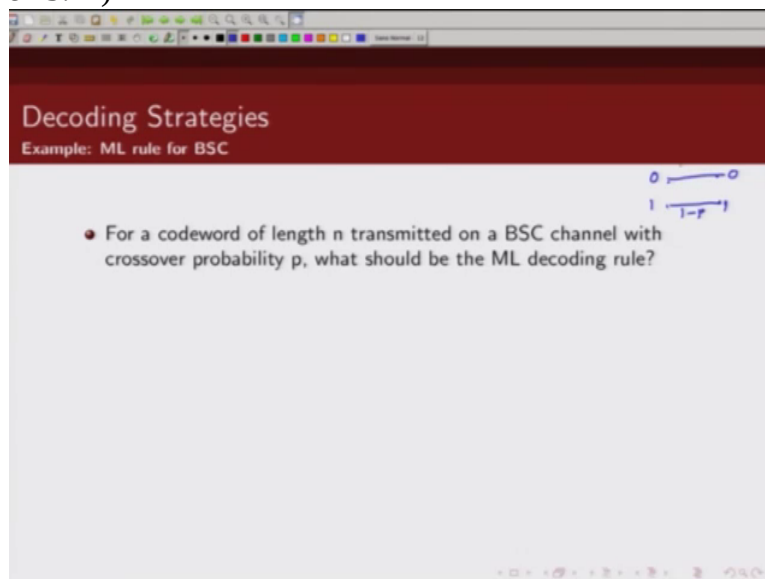
are interested in finding what would be the maximum likelihood decoding rule for a binary symmetric channel. Now recall what is a binary symmetric channel? There are two inputs;

(Refer Slide Time 23:03)



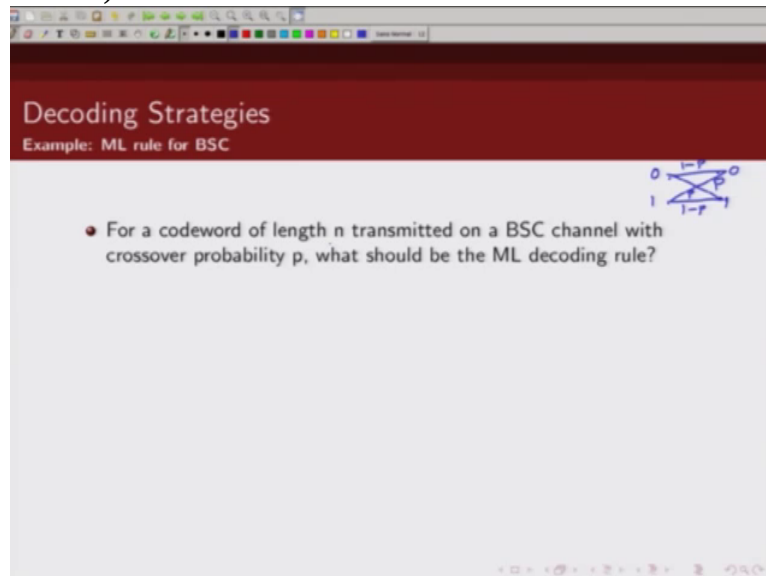
this is 0 and 1; two outputs 0 and 1 with probability 1 minus p.

(Refer Slide Time 23:11)



I received my bits correctly. And there is a crossover probability of p, Ok. So the question I am asking is, if I have a codeword of length n

(Refer Slide Time 23:28)



which is transmitted over a binary symmetric channel whose crossover probability is  $p$ , what should be my maximum likelihood decoding rule?

(Refer Slide Time 23:39)



So how do I solve it? As we just saw in the previous slide, maximizing probability of  $r$ , for maximum likelihood decoder we have to maximize probability of  $r$  given  $v$  which is equivalent to maximizing log of probability of  $r$  given  $v$ . So let's try to compute what's log of  $r$  given  $v$ , Ok? Now before I calculate

(Refer Slide Time 24:16)

Decoding Strategies  
Example: ML rule for BSC

- For a codeword of length  $n$  transmitted on a BSC channel with crossover probability  $p$ , what should be the ML decoding rule?
- Hamming distance  $d(\mathbf{r}, \mathbf{v})$  between  $\mathbf{r}$  and  $\mathbf{v}$  is the number of positions for which  $r_i \neq v_i$ .

$$\log P(\mathbf{r}/\mathbf{v}) = d(\mathbf{r}, \mathbf{v}) \log \left( \frac{p}{1-p} \right) + n \log(1-p)$$

probability of  $\mathbf{r}$  given  $\mathbf{v}$ , let me introduce another term which is called Hamming distance  
Now what is Hamming distance between two codewords? Hamming distance between two codewords or two  $n$  tuples, let's call it Hamming distance between  $\mathbf{r}$  and  $\mathbf{v}$ , both are  $n$  bit vector basically, so Hamming distance between  $\mathbf{r}$  and  $\mathbf{v}$  is defined as number of positions in which  $\mathbf{r}$  and  $\mathbf{v}$  are differing. So for example, if let's say  $\mathbf{r}$  is 1 1 1 0 1 1 and

(Refer Slide Time 24:55)

Decoding Strategies  
Example: ML rule for BSC

- For a codeword of length  $n$  transmitted on a BSC channel with crossover probability  $p$ , what should be the ML decoding rule?
- Hamming distance  $d(\mathbf{r}, \mathbf{v})$  between  $\mathbf{r}$  and  $\mathbf{v}$  is the number of positions for which  $r_i \neq v_i$ .

$$\log P(\mathbf{r}/\mathbf{v}) = d(\mathbf{r}, \mathbf{v}) \log \left( \frac{p}{1-p} \right) + n \log(1-p)$$

*r = 111011*

$\mathbf{v}$  is 0 1 1 1 0 1 then

(Refer Slide Time 24:59)

Decoding Strategies  
Example: ML rule for BSC

- For a codeword of length  $n$  transmitted on a BSC channel with crossover probability  $p$ , what should be the ML decoding rule?
- Hamming distance  $d(\mathbf{r}, \mathbf{v})$  between  $\mathbf{r}$  and  $\mathbf{v}$  is the number of positions for which  $r_i \neq v_i$ .

$$\log P(\mathbf{r}/\mathbf{v}) = d(\mathbf{r}, \mathbf{v}) \log \left( \frac{p}{1-p} \right) + n \log(1-p)$$

$r = 111011$   
 $v = 011101$

what is the Hamming distance? It's differing in the first location, 1, it is not differing here, not differing here, it is differing here, that's two, that's differing in this location. That's three. It's not differing in this location. So  $\mathbf{r}$  and  $\mathbf{v}$  differs in three locations, one is this location, other is this location. Third is this location. So the Hamming distance between  $\mathbf{r}$  and  $\mathbf{v}$  is

(Refer Slide Time 25:26)

Decoding Strategies  
Example: ML rule for BSC

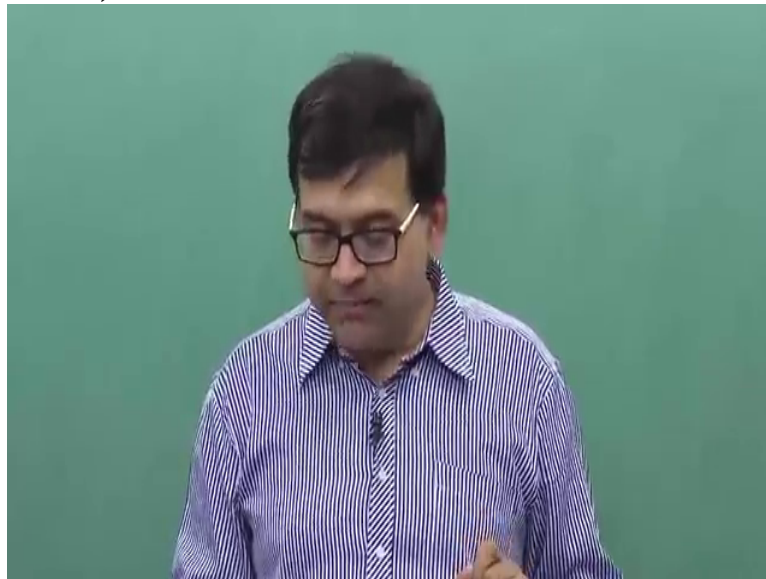
- For a codeword of length  $n$  transmitted on a BSC channel with crossover probability  $p$ , what should be the ML decoding rule?
- Hamming distance  $d(\mathbf{r}, \mathbf{v})$  between  $\mathbf{r}$  and  $\mathbf{v}$  is the number of positions for which  $r_i \neq v_i$ .

$$\log P(\mathbf{r}/\mathbf{v}) = d(\mathbf{r}, \mathbf{v}) \log \left( \frac{p}{1-p} \right) + n \log(1-p)$$

$r = 111011$   
 $v = 011101$

3 in this case Ok; now when we are sending a  $n$ -bit codeword over a binary symmetric channel, what happens?

(Refer Slide Time 25:38)



Some of the bits get flipped with probability, crossover probability  $p$ . Let's denote those number of flip bits by  $d$ .

(Refer Slide Time 25:51)

A presentation slide with a red header and white body. The header contains the text "Decoding Strategies" and "Example: ML rule for BSC". The body contains two bullet points, a mathematical formula, and handwritten binary strings. The first bullet point asks for the ML decoding rule for a codeword of length  $n$  on a BSC channel with crossover probability  $p$ . The second bullet point defines Hamming distance  $d(\mathbf{r}, \mathbf{v})$  as the number of positions where  $r_i \neq v_i$ . The formula is  $\log P(\mathbf{r}/\mathbf{v}) = d(\mathbf{r}, \mathbf{v}) \log \left( \frac{p}{1-p} \right) + n \log(1-p)$ . Handwritten in blue ink are the binary strings  $\mathbf{r} = 111011$  and  $\mathbf{v} = 011101$ , with a horizontal line under the last two bits of  $\mathbf{v}$ .

(Refer Slide Time 25:52)

Decoding Strategies  
Example: ML rule for BSC

- For a codeword of length  $n$  transmitted on a BSC channel with crossover probability  $p$ , what should be the ML decoding rule?
- Hamming distance  $d(\mathbf{r}, \mathbf{v})$  between  $\mathbf{r}$  and  $\mathbf{v}$  is the number of positions for which  $r_i \neq v_i$ .

$$\log P(\mathbf{r}/\mathbf{v}) = \underline{d(\mathbf{r}, \mathbf{v})} \log \left( \frac{p}{1-p} \right) + n \log(1-p)$$

$r = 111011$   
 $v = 011101$

So Hamming distance between two  $\mathbf{r}$  and  $\mathbf{v}$  will specify the locations where  $\mathbf{r}$  and  $\mathbf{v}$  are not same. When  $\mathbf{r}$  and  $\mathbf{v}$  are not same, that means those are locations where error has occurred. So number of positions that got flipped as a result of sending this codeword of binary symmetric channel, that is denoted by this  $d$  of  $\mathbf{r}$  and  $\mathbf{v}$ . And the remaining number of bits which did not get changed, that is basically  $n$  minus  $d$  of  $\mathbf{r}$  and  $\mathbf{v}$ . So these many bits

(Refer Slide Time 26:35)

Decoding Strategies  
Example: ML rule for BSC

- For a codeword of length  $n$  transmitted on a BSC channel with crossover probability  $p$ , what should be the ML decoding rule?
- Hamming distance  $d(\mathbf{r}, \mathbf{v})$  between  $\mathbf{r}$  and  $\mathbf{v}$  is the number of positions for which  $r_i \neq v_i$ .

$$\log P(\mathbf{r}/\mathbf{v}) = \underline{d(\mathbf{r}, \mathbf{v})} \log \left( \frac{p}{1-p} \right) + n \log(1-p)$$

$r = 111011$   
 $v = 011101$   
 $n - d(\mathbf{r}, \mathbf{v})$

did not get changed and these many bits got flipped.



(Refer Slide Time 26:40)

**Decoding Strategies**  
Example: ML rule for BSC

- For a codeword of length  $n$  transmitted on a BSC channel with crossover probability  $p$ , what should be the ML decoding rule?
- Hamming distance  $d(\mathbf{r}, \mathbf{v})$  between  $\mathbf{r}$  and  $\mathbf{v}$  is the number of positions for which  $r_i \neq v_i$ .

$$\log P(\mathbf{r}/\mathbf{v}) = d(\mathbf{r}, \mathbf{v}) \log \left( \frac{p}{1-p} \right) + n \log(1-p)$$

Handwritten examples:  
 $\mathbf{r} = 111011$   
 $\mathbf{v} = 011101$   
 $\frac{n - d(\mathbf{r}, \mathbf{v})}{d(\mathbf{r}, \mathbf{v})}$

So what is the probability that  $d$  bits got flipped? That is given by  $p$  raised to power

(Refer Slide Time 26:52)

**Decoding Strategies**  
Example: ML rule for BSC

- For a codeword of length  $n$  transmitted on a BSC channel with crossover probability  $p$ , what should be the ML decoding rule?
- Hamming distance  $d(\mathbf{r}, \mathbf{v})$  between  $\mathbf{r}$  and  $\mathbf{v}$  is the number of positions for which  $r_i \neq v_i$ .

$$\log P(\mathbf{r}/\mathbf{v}) = d(\mathbf{r}, \mathbf{v}) \log \left( \frac{p}{1-p} \right) + n \log(1-p)$$

Handwritten examples:  
 $\mathbf{r} = 111011$   
 $\mathbf{v} = 011101$   
 $\frac{n - d(\mathbf{r}, \mathbf{v})}{p^{d(\mathbf{r}, \mathbf{v})}}$

Diagram showing a transition from 0 to 0 with probability  $1-p$ , 0 to 1 with probability  $p$ , 1 to 0 with probability  $p$ , and 1 to 1 with probability  $1-p$ .

$d(\mathbf{r}, \mathbf{v})$  and what's the probability that  $n - d$  bits were received correctly? That is given by  $(1-p)^{n-d}$ . So we can

(Refer Slide Time 27:08)

**Decoding Strategies**  
Example: ML rule for BSC

• For a codeword of length  $n$  transmitted on a BSC channel with crossover probability  $p$ , what should be the ML decoding rule?

• Hamming distance  $d(\mathbf{r}, \mathbf{v})$  between  $\mathbf{r}$  and  $\mathbf{v}$  is the number of positions for which  $r_i \neq v_i$ .

$$\log P(\mathbf{r}/\mathbf{v}) = d(\mathbf{r}, \mathbf{v}) \log \left( \frac{p}{1-p} \right) + n \log(1-p)$$

Handwritten notes on the slide:  
 Diagram: A BSC channel with input 0 and 1, and output 0 and 1. Transitions are labeled with probabilities  $1-p$  and  $p$ .  
 Binary strings:  $\mathbf{r} = 111011$ ,  $\mathbf{v} = 011101$ .  
 Simplified formula:  $(1-p)^{n-d(r,v)} p^{d(r,v)}$

(Refer Slide Time 27:09)

**Decoding Strategies**  
Example: ML rule for BSC

• For a codeword of length  $n$  transmitted on a BSC channel with crossover probability  $p$ , what should be the ML decoding rule?

• Hamming distance  $d(\mathbf{r}, \mathbf{v})$  between  $\mathbf{r}$  and  $\mathbf{v}$  is the number of positions for which  $r_i \neq v_i$ .

$$\log P(\mathbf{r}/\mathbf{v}) = d(\mathbf{r}, \mathbf{v}) \log \left( \frac{p}{1-p} \right) + n \log(1-p)$$

Handwritten notes on the slide:  
 Diagram: A BSC channel with input 0 and 1, and output 0 and 1. Transitions are labeled with probabilities  $1-p$  and  $p$ .  
 Binary strings:  $\mathbf{r} = 111011$ ,  $\mathbf{v} = 011101$ .  
 Simplified formula:  $(1-p)^{n-d(r,v)} p^{d(r,v)}$

write probability of  $\mathbf{r}$  given  $\mathbf{v}$  as  $p$  raised to power  $d$  into  $1 - p$  raised to power  $n - d$ . If we take log on both sides, then this will basically become  $n - d \log(1 - p) + d \log p$ . Now we take terms containing  $d$  out, so what we will get is  $d \log \frac{p}{1-p} + n \log(1 - p)$ . So to maximize this probability we have to choose our  $\mathbf{v}$  such that this is maximized. Now look closely at both of these terms. Let us first look at this term. Does

(Refer Slide Time 28:18)

**Decoding Strategies**  
Example: ML rule for BSC

- For a codeword of length  $n$  transmitted on a BSC channel with crossover probability  $p$ , what should be the ML decoding rule?
- Hamming distance  $d(\mathbf{r}, \mathbf{v})$  between  $\mathbf{r}$  and  $\mathbf{v}$  is the number of positions for which  $r_i \neq v_i$ .

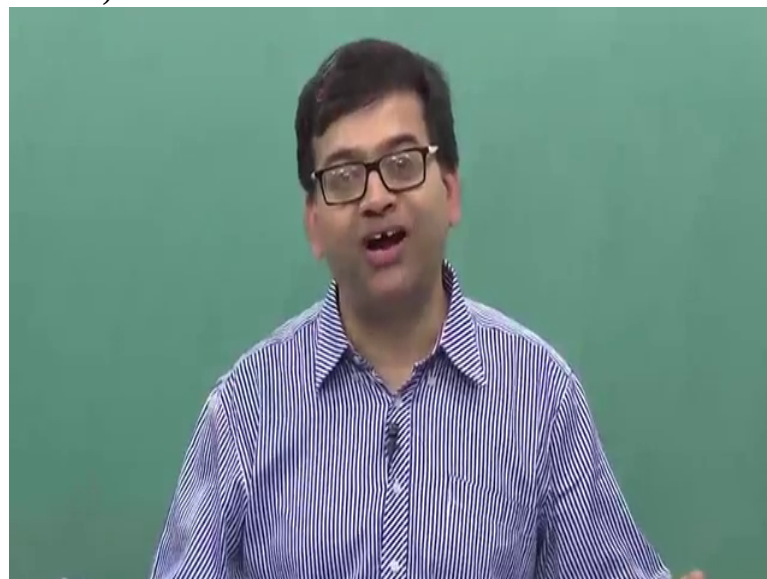
Diagram of a BSC channel: 0 and 1 inputs at the top, 0 and 1 outputs at the bottom. Transitions are labeled with  $1-p$  (correct) and  $p$  (crossover).

Handwritten examples:  $\mathbf{r} = 111011$ ,  $\mathbf{v} = 011101$

$$\log P(\mathbf{r}/\mathbf{v}) = d(\mathbf{r}, \mathbf{v}) \log \left( \frac{p}{1-p} \right) + \underbrace{n \log(1-p)}_{(1-p)^{n-d(\mathbf{r},\mathbf{v})} p^{d(\mathbf{r},\mathbf{v})}}$$

this term depend on selection of  $\mathbf{v}$ ? No. It depends on  $n$  which is codeword length. It depends on crossover probability  $p$ . So whatever  $\mathbf{v}$  we choose it does not change this probability. So in other words, to maximize this then, we will have to maximize this first term. Now look at this term closely. Typically the crossover probability will be smaller than

(Refer Slide Time 28:49)



half If that happens, what happens

(Refer Slide Time 28:54)

**Decoding Strategies**  
Example: ML rule for BSC

• For a codeword of length  $n$  transmitted on a BSC channel with crossover probability  $p$ , what should be the ML decoding rule?

• Hamming distance  $d(\mathbf{r}, \mathbf{v})$  between  $\mathbf{r}$  and  $\mathbf{v}$  is the number of positions for which  $r_i \neq v_i$ .

$\log P(\mathbf{r}/\mathbf{v}) = d(\mathbf{r}, \mathbf{v}) \log \left( \frac{p}{1-p} \right) + n \log(1-p)$

Handwritten notes on the slide:  
 Diagram: A square representing a BSC channel with input 0 and 1, and output 0 and 1. Transitions are labeled with  $1-p$  and  $p$ .  
 Equations:  $\mathbf{r} = 111011$ ,  $\mathbf{v} = 011101$ .  
 The equation above is annotated with  $(1-p)^{n-d(r,v)}$  and  $p^{d(r,v)}$ .

to this ratio,  $p$  by  $1 - p$ ? This will be some ratio between 0 and 1. And what happens to  $\log$  of a number which is between 0 and 1? That is a negative quantity. So what we get then is; to maximize this, we have to maximize minus of  $d(\mathbf{r}, \mathbf{v})$ , correct? So a maximum likelihood decoder will choose a  $\mathbf{v}$  such that minus of  $d(\mathbf{r}, \mathbf{v})$  is maximized. In other words, we should choose a codeword  $\mathbf{v}$  in such a way such that  $d(\mathbf{r}, \mathbf{v})$  is minimized. When  $d(\mathbf{r}, \mathbf{v})$  is minimized, then only minus of  $d(\mathbf{r}, \mathbf{v})$  will be maximized. So that's what

(Refer Slide Time 29:53)

**Decoding Strategies**  
Example: ML rule for BSC

• For a codeword of length  $n$  transmitted on a BSC channel with crossover probability  $p$ , what should be the ML decoding rule?

• Hamming distance  $d(\mathbf{r}, \mathbf{v})$  between  $\mathbf{r}$  and  $\mathbf{v}$  is the number of positions for which  $r_i \neq v_i$ .

$\log P(\mathbf{r}/\mathbf{v}) = d(\mathbf{r}, \mathbf{v}) \log \left( \frac{p}{1-p} \right) + n \log(1-p)$

• Note  $\log \left( \frac{p}{1-p} \right) < 0$  for  $p < 1/2$  and  $n \log(1-p)$  is a constant.

we are seeing here.  $\log$  of  $p / (1 - p)$  is less than zero so this will be a negative quantity. When you want to maximize a negative quantity, this term should be as small as possible. And this term does not depend on selection of  $\mathbf{v}$ .

(Refer Slide Time 30:13)

**Decoding Strategies**  
Example: ML rule for BSC

- For a codeword of length  $n$  transmitted on a BSC channel with crossover probability  $p$ , what should be the ML decoding rule?
- Hamming distance  $d(\mathbf{r}, \mathbf{v})$  between  $\mathbf{r}$  and  $\mathbf{v}$  is the number of positions for which  $r_i \neq v_i$ .

$$\log P(\mathbf{r}/\mathbf{v}) = d(\mathbf{r}, \mathbf{v}) \log \left( \frac{p}{1-p} \right) + n \log(1-p)$$

- Note  $\log \left( \frac{p}{1-p} \right) < 0$  for  $p < 1/2$  and  $n \log(1-p)$  is a constant.
- For each  $\mathbf{r}$ , choose  $\hat{\mathbf{v}}$  as the codeword  $\mathbf{v}$  which minimizes the Hamming distance  $d(\mathbf{r}, \mathbf{v})$ .

So this gives us a decoding, maximum likelihood decoding rule for binary symmetric channel. And what is that? We should choose a  $\mathbf{v}$  such that  $d(\mathbf{r}, \mathbf{v})$  is minimized. In other words we should choose a codeword  $\mathbf{v}$  such that Hamming distance between the codeword  $\mathbf{v}$  and the received sequences is minimized. And that makes sense. And that's our maximum likelihood decoding rule.

(Refer Slide Time 30:47)

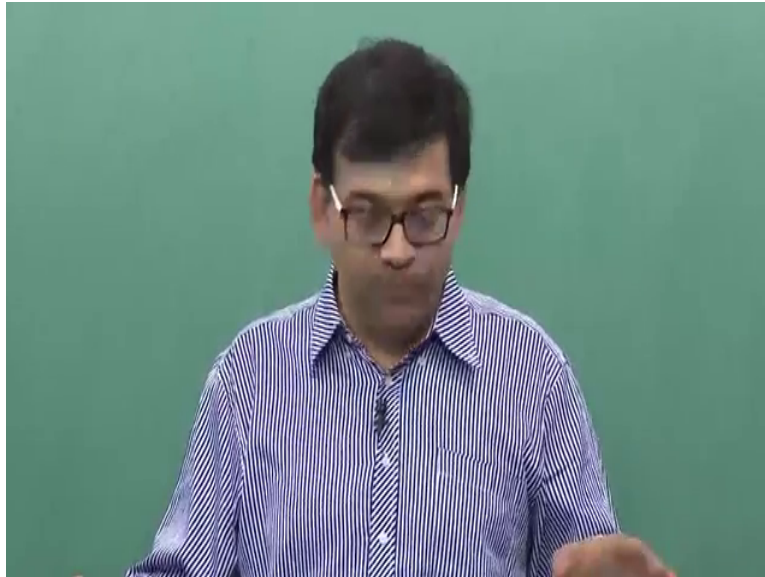
**Error control strategies**

Forward Error Correction (FEC):

- In *one-way* system, transmission takes place only in one direction, i.e. from transmitter to receiver.

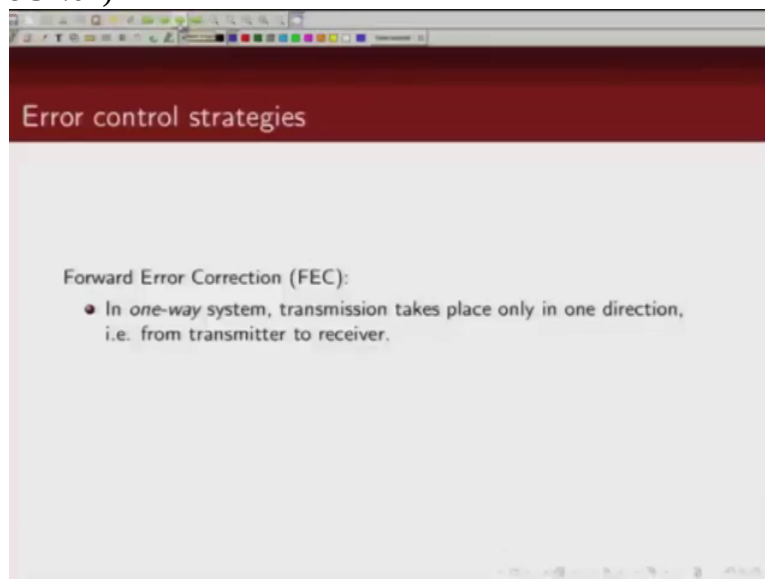
So finally I am going to conclude this lecture with definition of few

(Refer Slide Time 30:55)



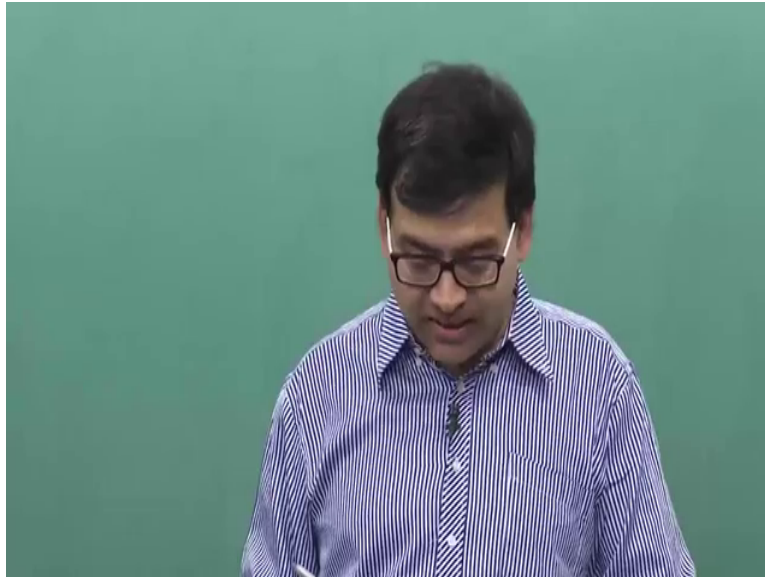
error correcting strategies The first one which I am going to describe

(Refer Slide Time 31:01)



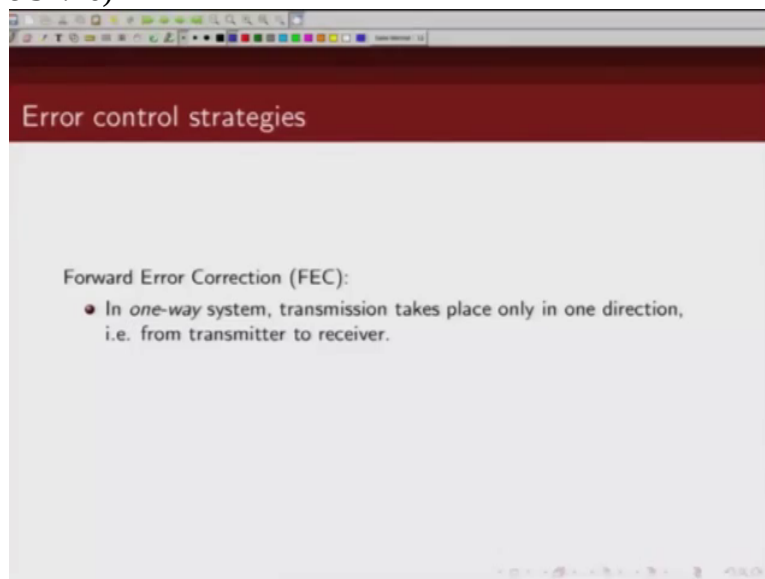
is what is known as F E C, forward error correction So in systems where there is no feedback from the receiver to the transmitter,

(Refer Slide Time 31:13)



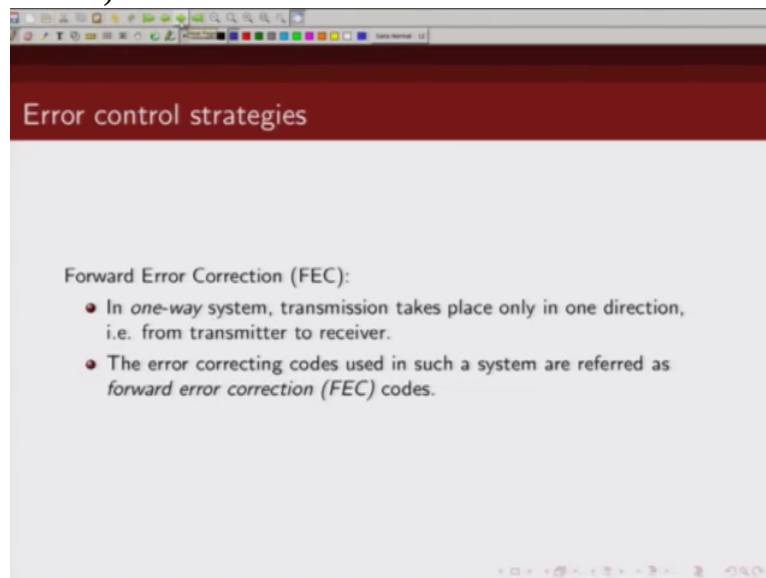
we are calling those systems as

(Refer Slide Time 31:16)



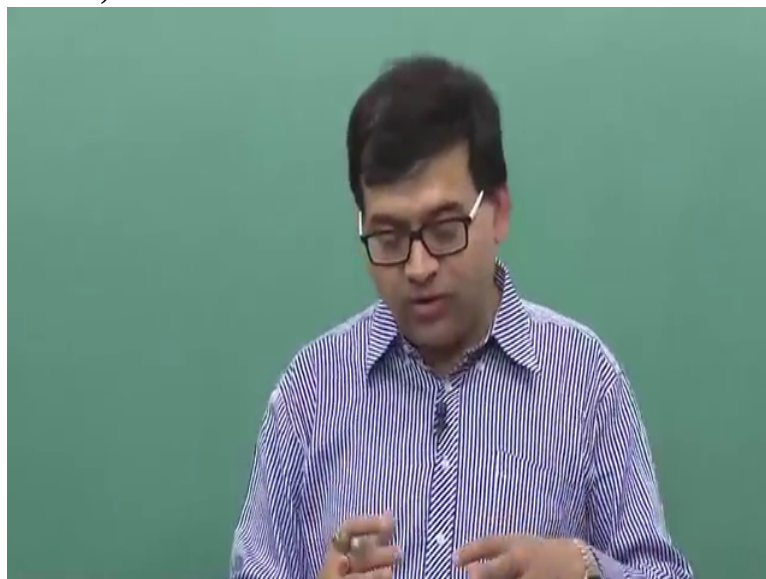
one way systems, where transmission happens only in one direction, from transmitter to receiver In those systems the error correcting codes

(Refer Slide Time 31:25)



that are used are known as F E C. So when you hear this term F E C code, it basically means basically when we are sending,

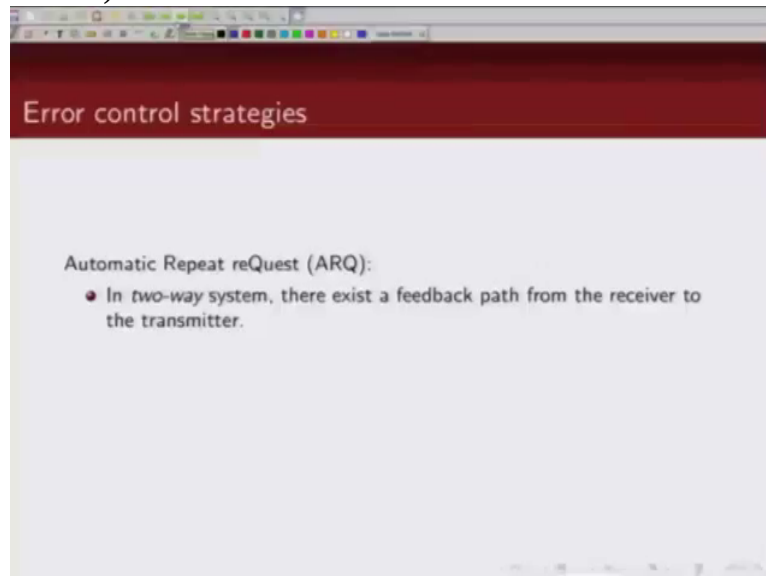
(Refer Slide Time 31:35)



so this is error correcting code used for, when we are using transmission one way, from transmitter to receiver Now in some cases, we have a mechanism of feedback from the receiver to back to the transmitter.

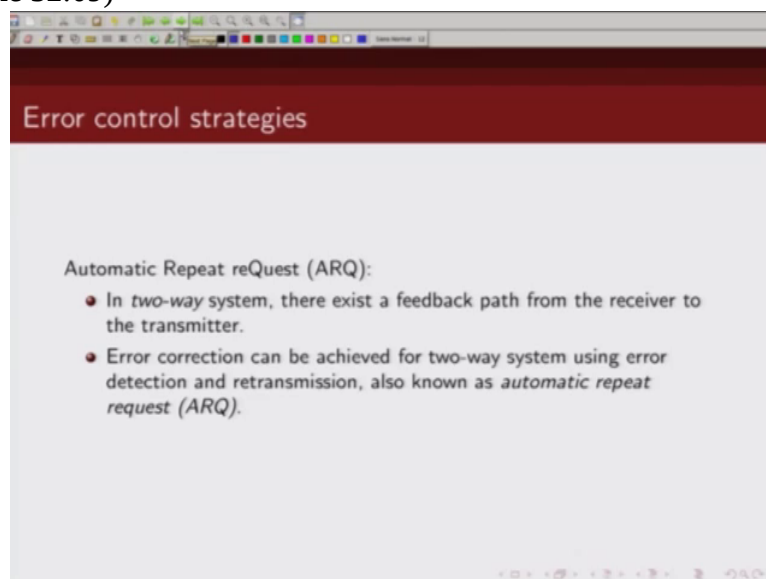


(Refer Slide Time 31:57)



So in those cases where there exists a feedback from receiver to the transmitter we are calling these systems as two way systems. So for these systems basically

(Refer Slide Time 32:09)



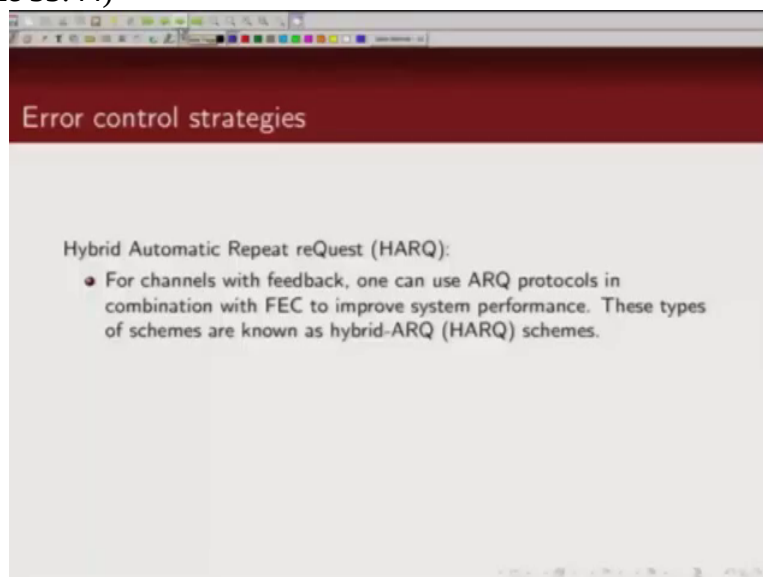
the error correcting strategy which is used is what is known as automated repeat request. Now how does this work? So you initially send some coded

(Refer Slide Time 32:22)



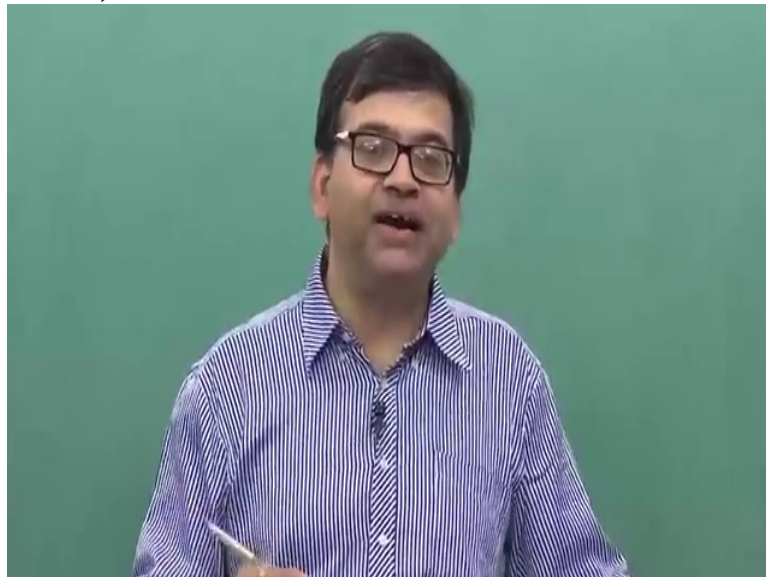
packets where you just use parity bits for error detection. So you send your information bits and some few parity bits for error detection. At the receiver, using those parity bits, the receiver will try to judge whether there is any error in the received packet. If it finds that there are errors, it will send a negative acknowledgement and again you will retransmit the same packet or some additional parity bits. So that's basically same packet, basically. So that is your automatic repeat request scheme. Now in this automatic repeat request scheme, the idea is you are sending an uncoded packet with some bits for error detection. So you are not really sending any bits for error correction. So only, so this is typically useful if the links are very good. You just are sending some uncoded packets with some bits for error detection and occasionally when the packets are not received correctly then you ask for re-transmission.

(Refer Slide Time 33:44)



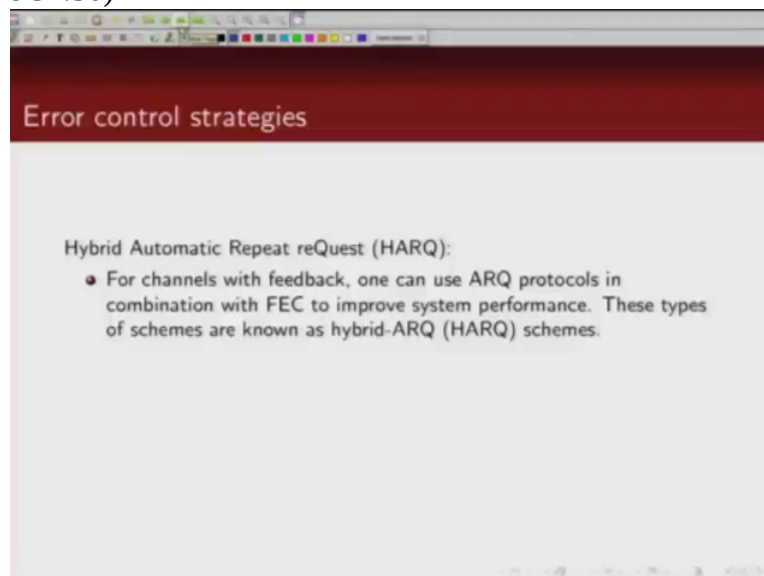
A strategy that combines both forward error correction and a r q is known as hybrid a r q

(Refer Slide Time 33:51)



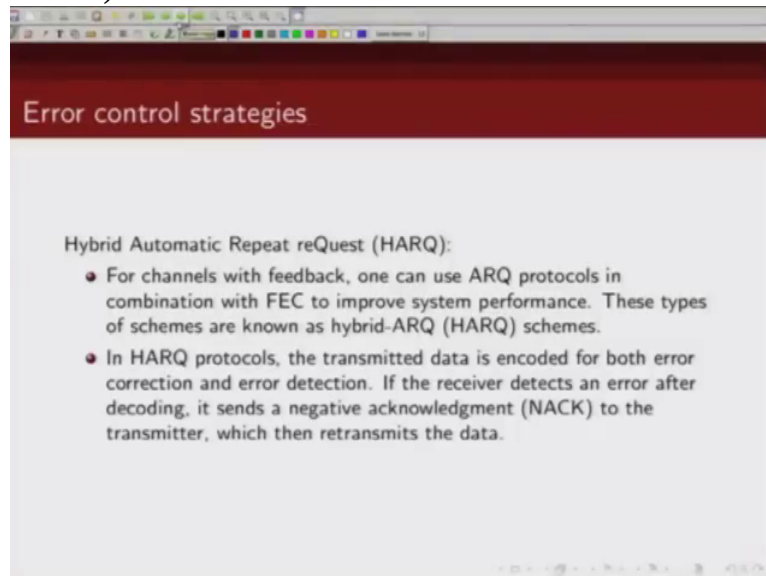
In this, you send coded packets from transmitter to receiver. Now if these coded packets are not received correctly by the receiver, the receiver will basically send a negative acknowledgment and then you will send, resend the same packet or you will try to send some additional parity bits and using those additional parity bits you will try to now decode the original packet. So hybrid a r q is a combination of forward error correcting scheme and automatic repeat request scheme. Typically

(Refer Slide Time 34:30)



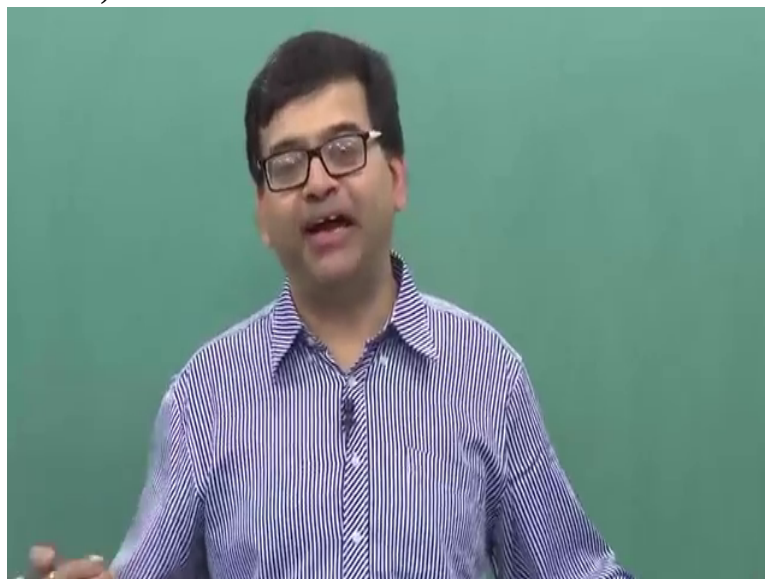
it's in a communication

(Refer Slide Time 34:32)



system, you will see a combination of both forward error correcting schemes and hybrid a r q schemes used.

(Refer Slide Time 34:38)



So with this, I am going to conclude this lecture. Thank you