

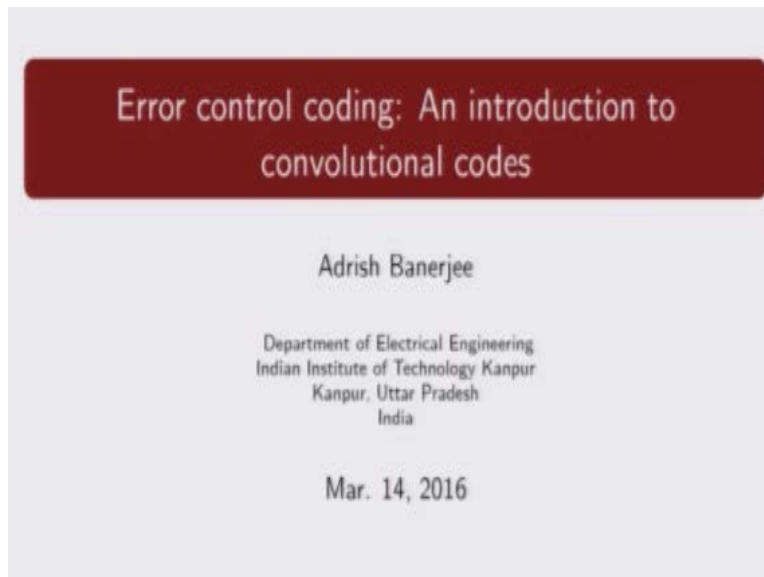
**Indian Institute of Technology Kanpur**  
**National Programme on Technology Enhanced Learning (NPTEL)**  
**Course Title**  
**Error Control Coding: An Introduction to Convolutional Codes**

**Lecture – 1C**  
**Introduction to Error Control Coding – III**

**by**  
**Prof. Adrish Banerjee**  
**Department of Electrical Engineering, IIT Kanpur**

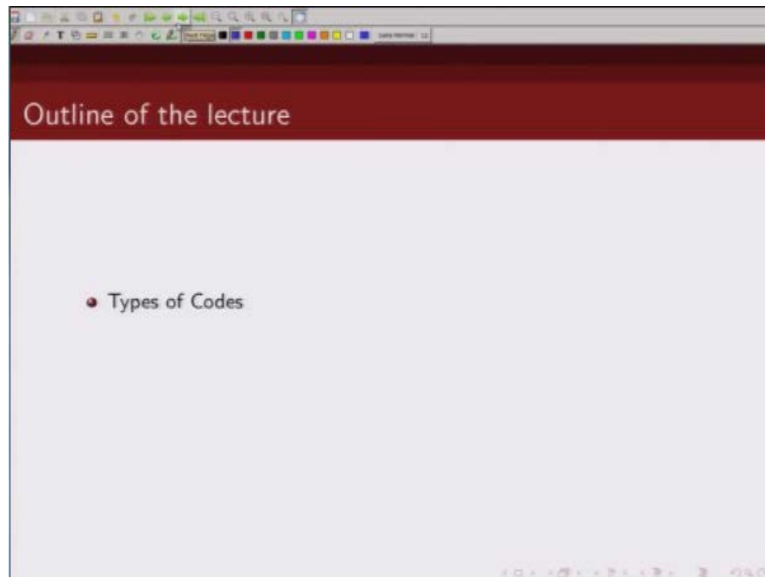
Welcome to course on Error Control Coding, an introduction to convolutional code.

(Refer Slide Time: 00:19)



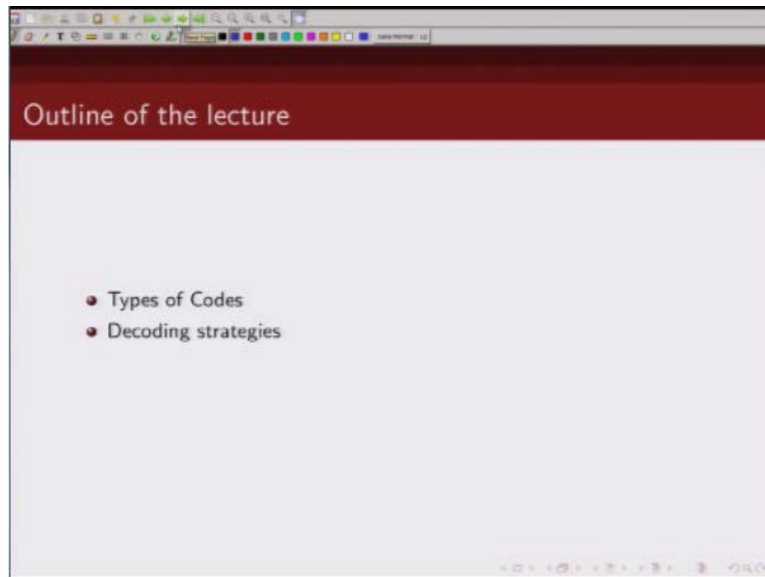
So in this lecture we will conclude basically our discussions on introduction. We are going to talk about the differences between a block code and a convolutional code, then we are going to talk about, how should we design our decoding strategy for correcting error control coding. And then finally we are going to describe what we mean by forward error correction and hybrid automatic repeat request system.

(Refer Slide Time: 00:53)



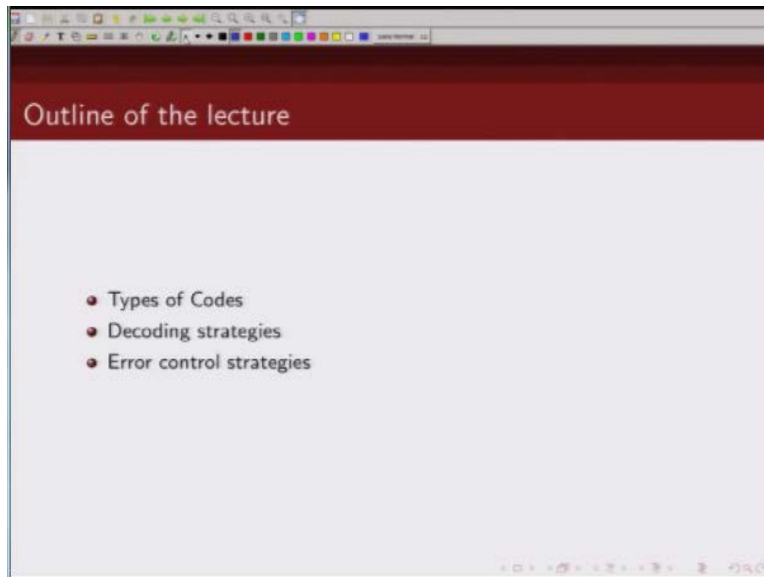
So as I said we will first describe – so error correcting codes can be broadly classified into two classes, block codes and convolutional codes. We will describe what is meant by block code and what is meant by convolutional code, and we will bring out a difference and similarities between the two.

(Refer Slide Time: 01:16)



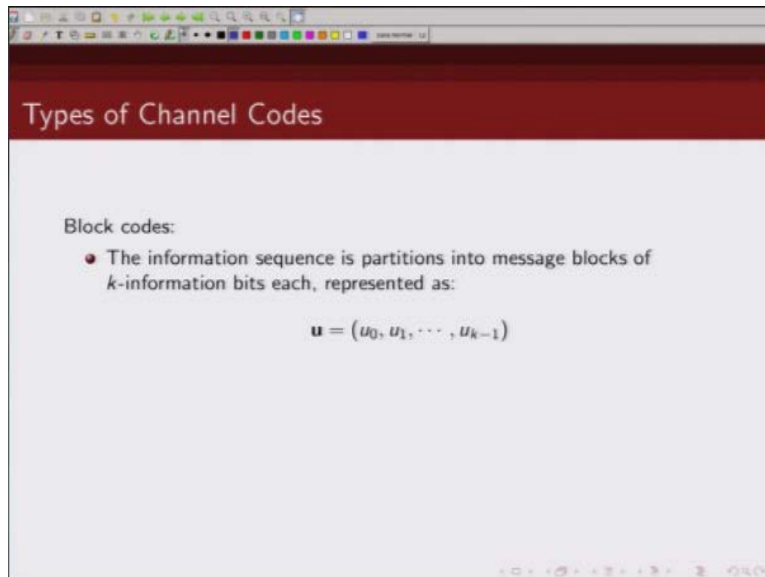
Then we will talk about various decoding strategies.

(Refer Slide Time: 01:19)



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

(Refer Slide Time: 01:29)



Types of Channel Codes

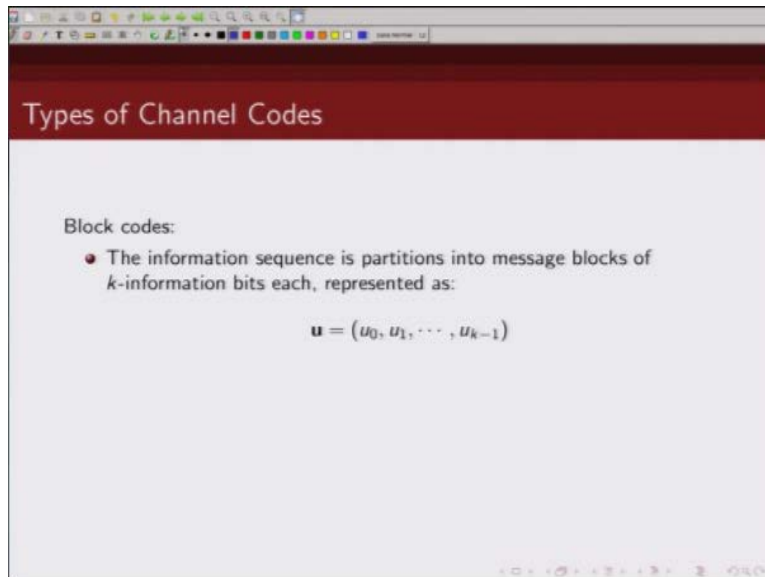
Block codes:

- The information sequence is partitioned into message blocks of  $k$ -information bits each, represented as:

$$\mathbf{u} = (u_0, u_1, \dots, u_{k-1})$$

So let us start with what is block codes, so as the name suggest in block codes we take a block of  $k$ -bits and map it to an  $n$ -bit code word.

(Refer Slide Time: 01:49)



Types of Channel Codes

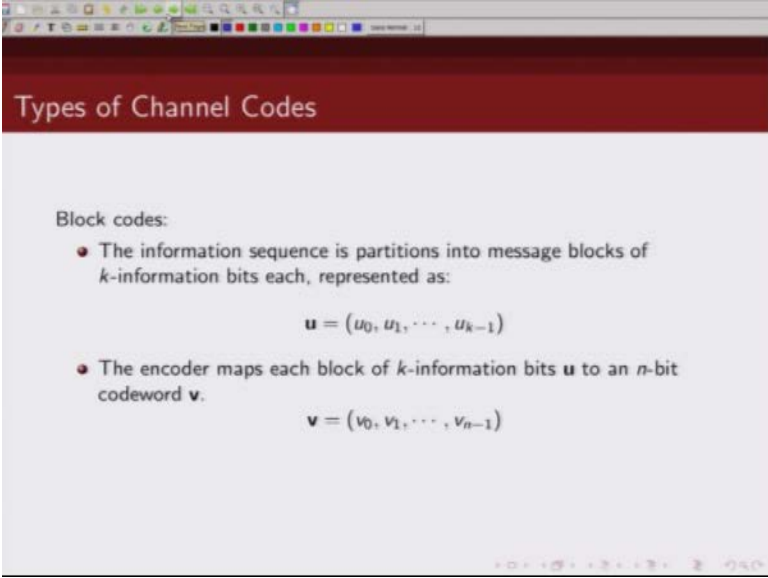
Block codes:

- The information sequence is partitioned into message blocks of  $k$ -information bits each, represented as:

$$\mathbf{u} = (u_0, u_1, \dots, u_{k-1})$$

So information sequence is passed into blocks of  $k$ -bits, and we take this block of  $k$ -bits and map it to a block of  $n$ -bits.

(Refer Slide Time: 01:59)



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  $\mathbf{v}$ .  
$$\mathbf{v} = (v_0, v_1, \dots, v_{n-1})$$

So 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  $n$ -code are is going to map this  $k$ -bits into an  $n$ -bit sequence which is denoted by  $\mathbf{v}$ .

(Refer Slide Time: 02:19)

Types of Channel Codes

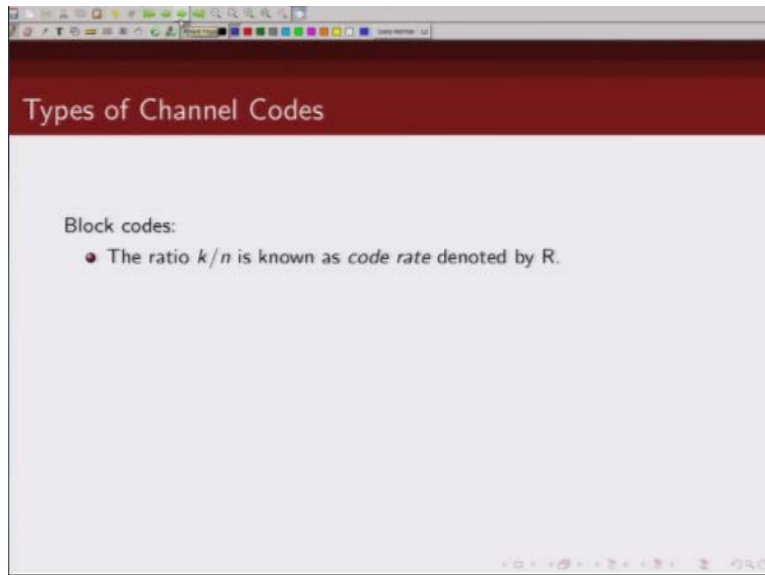
Block codes:

- 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.

Now in block codes the encoder is memoryless, what do we mean by that, so when we encode a block of  $k$ -bits our output depends only on that current block of  $k$ -bits, it does not depend on what were 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.

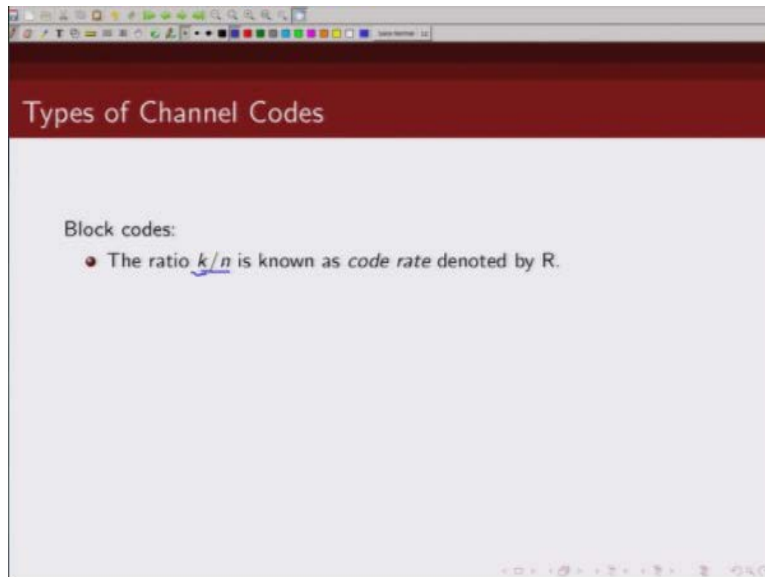


(Refer Slide Time: 02:56)



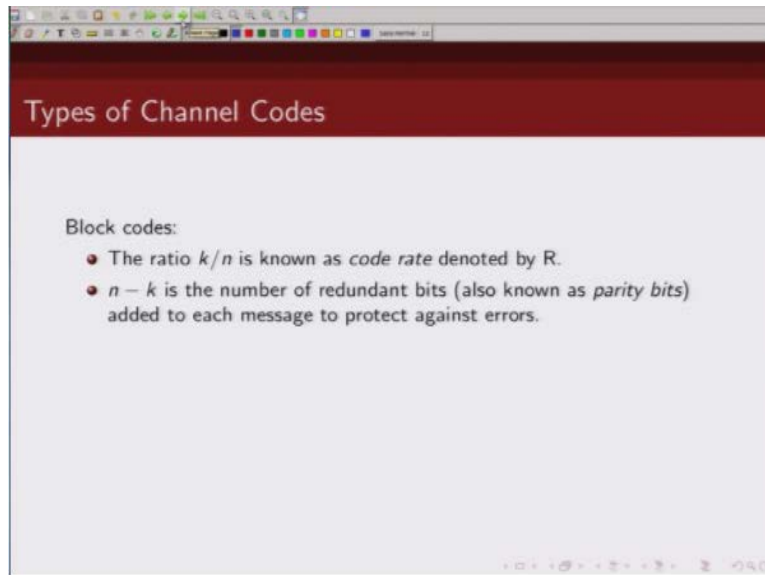
As we mentioned in the previous lectures, we defined our code rate.

(Refer Slide Time: 03:02)



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.

(Refer Slide Time: 03:17)



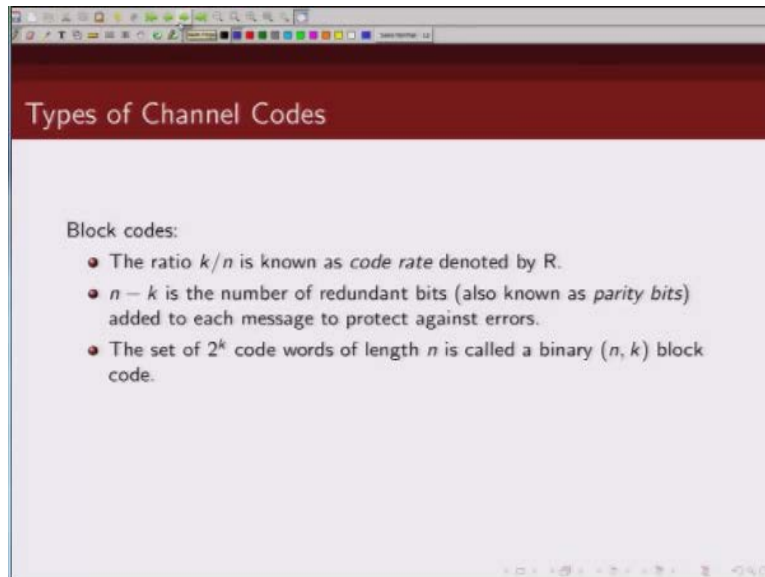
Types of Channel Codes

Block codes:

- The ratio  $k/n$  is known as *code rate* denoted by  $R$ .
- $n - k$  is the number of redundant bits (also known as *parity bits*) added to each message to protect against errors.

And it is typically denoted by  $r$ .  $K$  is number of information bits;  $n$  is a number of coded bits. So  $n-k$  is number of redundant bits that we are adding to our information bits. And these are also known as parity bits.

(Refer Slide Time: 03:36)



Types of Channel Codes

Block codes:

- The ratio  $k/n$  is known as *code rate* denoted by  $R$ .
- $n - k$  is the number of redundant bits (also known as *parity bits*) added to each message to protect against errors.
- The set of  $2^k$  code words of length  $n$  is called a binary  $(n, k)$  block code.

If we are considering without loss of generality we will basically consider in these set of lectures binary code word, so our information sequence consist of zeros and ones. Similarly our code sequence also consists of zeros and ones. Since we are considering a block of  $k$ -bits and binary code words, so number of code words is basically  $2^k$ .

(Refer Slide Time: 04:04)

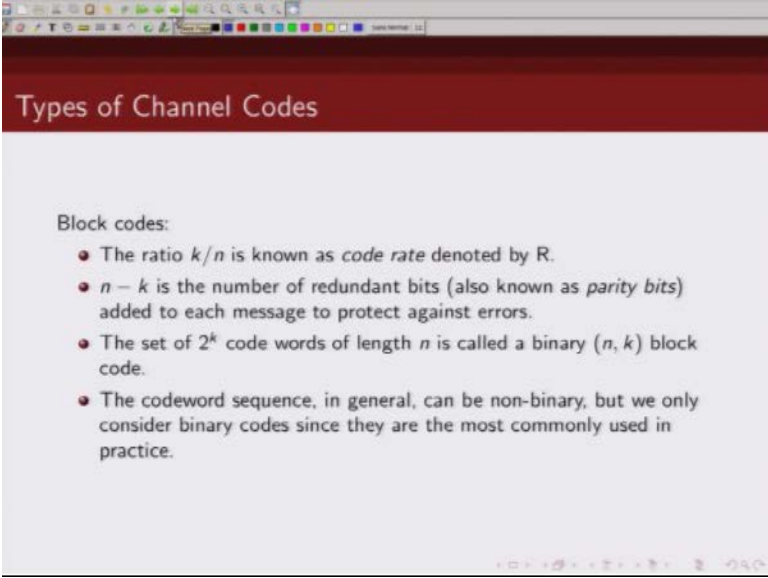
Types of Channel Codes

Block codes:

- The ratio  $k/n$  is known as *code rate* denoted by  $R$ .
- $n - k$  is the number of redundant bits (also known as *parity bits*) added to each message to protect against errors.
- The set of  $2^k$  code words of length  $n$  is called a binary  $(n, k)$  block code.

So a binary  $(n, k)$  block code consist of  $2^k$  code words each of length  $n$ .

(Refer Slide Time: 04:12)



The image shows a presentation slide with a dark red header containing the title "Types of Channel Codes". Below the header, the text "Block codes:" is followed by a bulleted list of four points. The slide is displayed within a window that has a standard operating system taskbar at the top and a navigation bar at the bottom.

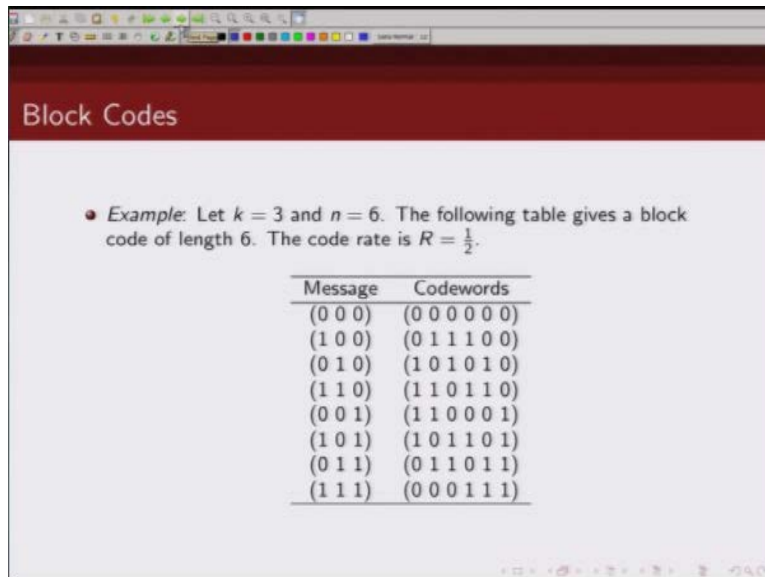
Types of Channel Codes

Block codes:

- The ratio  $k/n$  is known as *code rate* denoted by  $R$ .
- $n - k$  is the number of redundant bits (also known as *parity bits*) added to each message to protect against errors.
- The set of  $2^k$  code words of length  $n$  is called a binary  $(n, k)$  block code.
- The codeword sequence, in general, can be non-binary, but we only consider binary codes since they are the most commonly used in practice.

Now these code words need not be binary, however it is a same story mostly applies to non-binary code words as well, so we will restrict our discussion to binary code words.

(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 let us consider an example of a linear block code, so in this example a number of information bits is three, the number of coded bits is six. So the code rate which is ratio of number of information bits to number of coded bits is  $\frac{3}{6}$  which is  $\frac{1}{2}$ .

(Refer Slide Time: 04:51)

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  $k=3$  that means there are  $2^3$  which is eight code words. And these are basically from 000 to 111; these are the eight code words. Now the message, these are the eight message bits and corresponding to these message bits, these are the eight code words. Now 000 is mapped to all the zero sequence, 100 is mapped to (011100) 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 code word. So let us look at each of the columns of these code words, so let us look at this column first which is (00001111). So how was this column, how did we map to get this column, if we look at information bits, this column is nothing but same as this information bit, you can see (00001111).

Similarly look at this one, this column is same as this column (00110011) and this column is same as this column. So in other words this bit of the code word is same as this bit of the information sequence, this bit of the code word is same as this bit of the information sequence, this bit of the code word is same as this bit of the information sequence. Now let us look at this one.



So if we do a XOR of these two, look at this  $0 + 0 = 0$ ,  $1 + 0 = 1$ ,  $0 + 1 = 1$ ,  $1 + 1 = 0$ , we are talking about binary. Addition nowhere binary field so,  $0 + 0 = 0$ ,  $0 + 1 = 1$ ,  $1 + 0 = 1$  and  $1 + 1 = 0$ , it is modular to addition. So  $1 + 1 = 0$ , this is  $0 + 0 = 0$ ,  $1 + 0 = 1$ ,  $0 + 1 = 1$  and  $1 + 1 = 0$ .

(Refer Slide Time: 07:53)

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 – let us say denote this by  $U_0, U_1, U_2$  and we denote these by  $V_0, V_1, V_2, V_3, V_4$  and  $V_5$ , what we have found out so far is  $V_5$  is same as  $U_2$ ,  $V_4$  is same as  $U_1$ ,  $V_3$  is same as  $U_0$  and what is  $V_2$ ?  $V_2$  was  $U_0 + U_1$ . Now let us look at  $V_1$ , if we look at these two  $U_0 + U_2$ , so  $0 + 0 = 0$ ,  $1 + 0 = 1$ ,  $0 + 0 = 0$ ,  $1 + 0 = 1$ ,  $0 + 1 = 1$ ,  $1 + 1 = 0$ ,  $0 + 1 = 1$  and  $1 + 1 = 0$ .

So  $V_1$  is nothing but  $U_0 + U_2$  okay. Now look at last, this one  $V_0$ , what is  $V_0$ ? We can see that this is same as  $U_1 + U_2$ .

(Refer Slide Time: 09:30)

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

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

This is  $U_1 + U_2$ , so  $0 + 0 = 0, 0 + 0 = 0, 1 + 0 = 1, 1 + 0 = 1, 0 + 1 = 1, 0 + 1 = 1, 1 + 1 = 0$  and  $1 + 1 = 0$ .

(Refer Slide Time: 09:48)

**Block Codes**

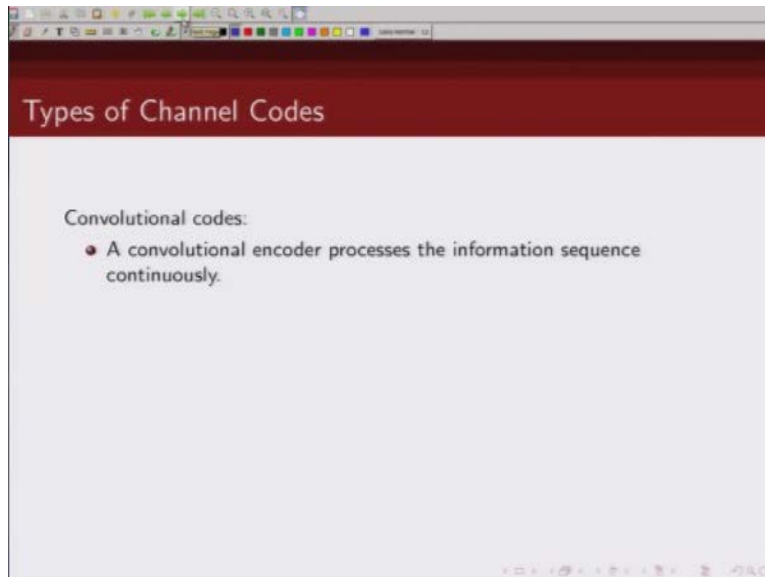
• Example: Let  $k = 3$  and  $n = 6$ . The following table gives a block code of length 6. The code rate is  $R = \frac{1}{2}$ .

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)

$v_5 = v_2$   
 $v_4 = v_1$   
 $v_3 = v_0$   
 $v_2 = v_0 + v_1$   
 $v_1 = v_0 + v_2$   
 $v_0 = v_1 + v_2$

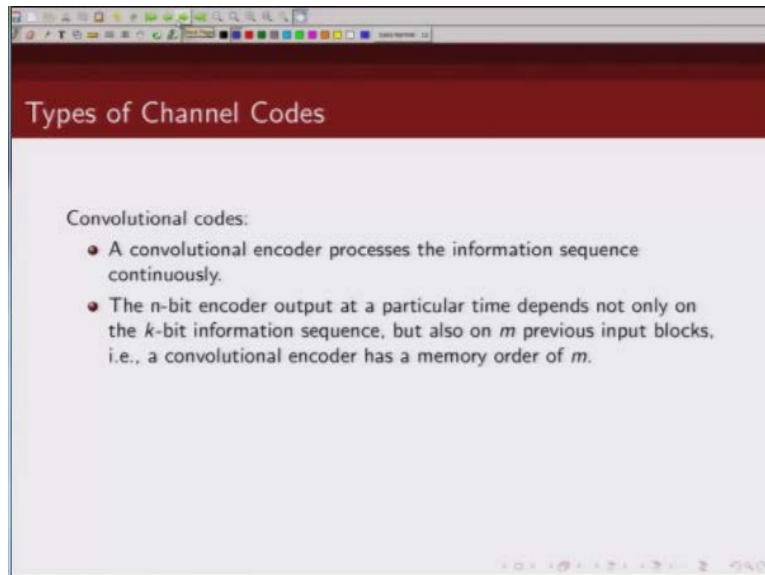
So this is how we have mapped our information bits into our coded bits okay. So again to recap in block codes we take, we partition our information sequence into blocks of  $k$ -bits and we map these  $k$ -bits into blocks of  $n$ -bits, and this mapping is memory less. In other words how we map these  $k$ -bits does not depend on how we have mapped the previous blocks of  $k$ -bits.

(Refer Slide Time: 10:28)



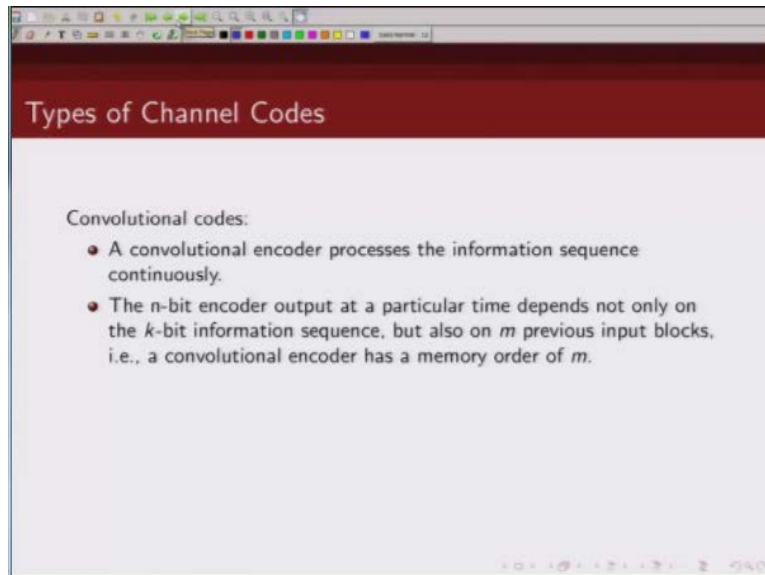
Okay, so let us now contrast it with what are convolutional codes and how are they different from convolutional codes. So in block codes we are for information sequence into blocks of data and we handle them block by block, whereas in a convolutional code you can process your information sequence in a continuous fashion.

(Refer Slide Time: 10:53)



The second difference is the encoding in convolutional code is with memory. In other words the current output not only depends on current input, but it also depends on past inputs and outputs okay. So unlike block codes in convolutional codes output depends on past inputs and outputs.

(Refer Slide Time: 11:24)



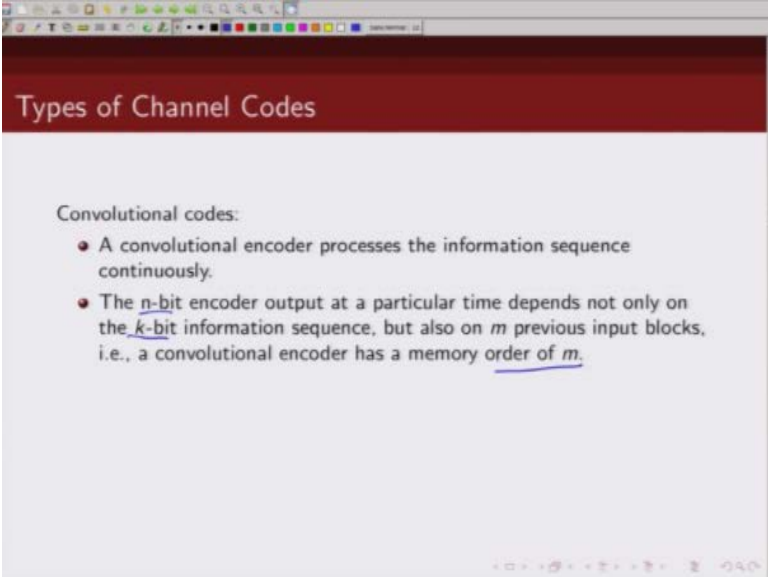
Types of Channel Codes

Convolutional codes:

- A convolutional encoder processes the information sequence continuously.
- The  $n$ -bit encoder output at a particular time depends not only on the  $k$ -bit information sequence, but also on  $m$  previous input blocks, i.e., a convolutional encoder has a memory order of  $m$ .

So if we have an  $(n, k)$  convolutional codes where  $k$  is a number of information bits,  $n$  is the number of coded bits, we have another parameter we are calling it memory order which signifies basically how many past bits or how many, what is the past information that is being used –

(Refer Slide Time: 11:48)



The image shows a screenshot of a presentation slide. The slide has a dark red header with the title "Types of Channel Codes" in white text. Below the header, the text "Convolutional codes:" is followed by two bullet points. The first bullet point states that a convolutional encoder processes the information sequence continuously. The second bullet point states that the  $n$ -bit encoder output at a particular time depends not only on the  $k$ -bit information sequence, but also on  $m$  previous input blocks, and that a convolutional encoder has a memory order of  $m$ . The words "k-bit" and "memory order of m" are underlined in the original image. The slide also features a standard Windows taskbar at the top and a navigation bar at the bottom.

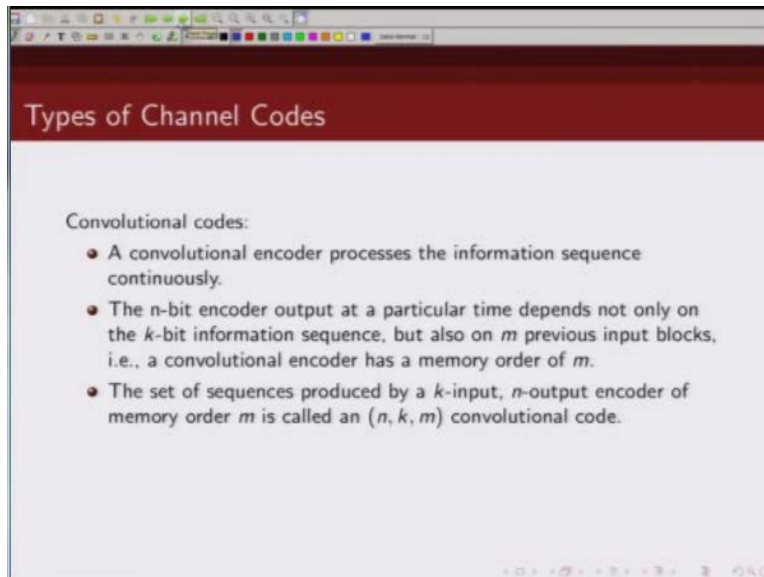
Types of Channel Codes

Convolutional codes:

- A convolutional encoder processes the information sequence continuously.
- The  $n$ -bit encoder output at a particular time depends not only on the  $k$ -bit information sequence, but also on  $m$  previous input blocks, i.e., a convolutional encoder has a memory order of  $m$ .

To generate their current output.

(Refer Slide Time: 11:55)



Types of Channel Codes

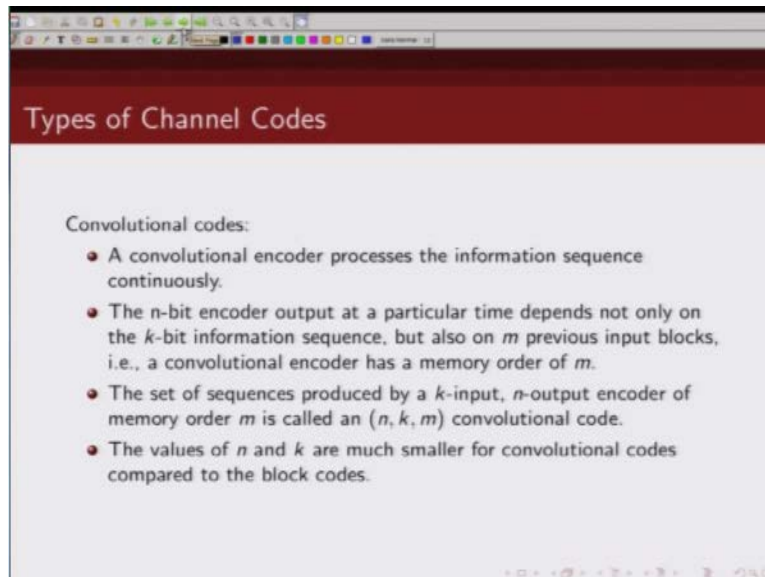
Convolutional codes:

- A convolutional encoder processes the information sequence continuously.
- The  $n$ -bit encoder output at a particular time depends not only on the  $k$ -bit information sequence, but also on  $m$  previous input blocks, i.e., a convolutional encoder has a memory order of  $m$ .
- The set of sequences produced by a  $k$ -input,  $n$ -output encoder of memory order  $m$  is called an  $(n, k, m)$  convolutional code.

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



(Refer Slide Time: 12:12)



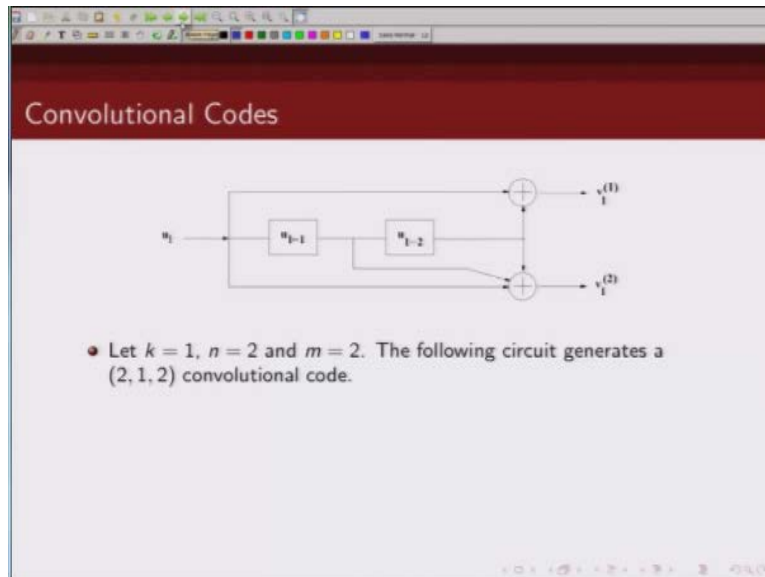
### Types of Channel Codes

Convolutional codes:

- A convolutional encoder processes the information sequence continuously.
- The  $n$ -bit encoder output at a particular time depends not only on the  $k$ -bit information sequence, but also on  $m$  previous input blocks, i.e., a convolutional encoder has a memory order of  $m$ .
- The set of sequences produced by a  $k$ -input,  $n$ -output encoder of memory order  $m$  is called an  $(n, k, m)$  convolutional code.
- The values of  $n$  and  $k$  are much smaller for convolutional codes compared to the block codes.

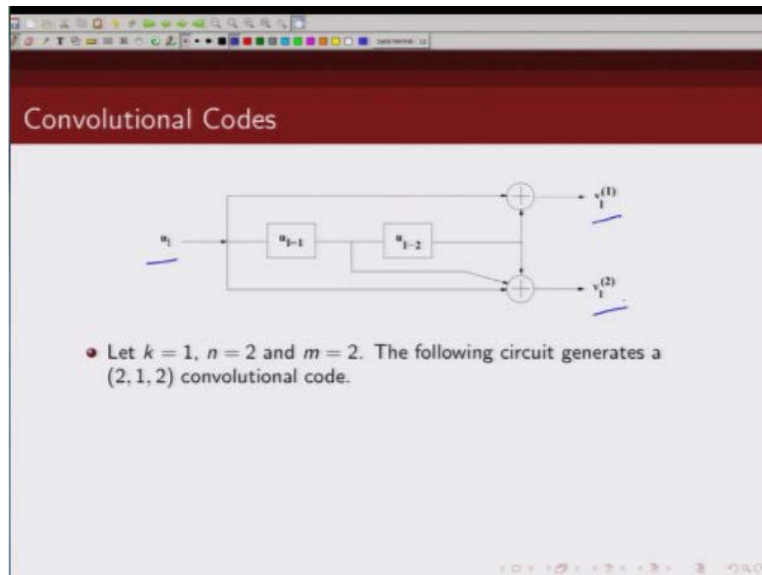
Another certain difference in case of convolutional codes typically the values of  $k$  and  $n$  are much smaller compared to values of  $k$  and  $n$  for block codes.

(Refer Slide Time: 12:28)



So let us think an example now for convolutional codes, so here we have one input and two outputs.

(Refer Slide Time: 12:36)



The input we are denoting by  $u(i)$ , output we are denoting by  $v(i^1)$ ,  $v(i^2)$ . Now note here each of the outputs here, not only depends on the current input, because you want, but it also depends on these past values, it also depends on what  $u_{i-1}$  was, what  $u_{i-2}$  was. So this is an example of memory order to, 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,  $n = 2$ , there are two outputs,  $k = 1$ , one input and memory order 2, because the output depends on past two values of information sequence.

(Refer Slide Time: 13:40)

### Convolutional Codes

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

$$v_i^{(1)} = u_i + u_{i-2}$$
$$v_i^{(2)} = u_i + u_{i-1} + u_{i-2}$$

So you can see here, the first input which is  $v_1$ ,  $v_1^1$  it is basically  $u_1 + u_{1-2}$ , see 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 then this  $u_{1-2}$ . So this is basically how --

(Refer Slide Time: 14:16)

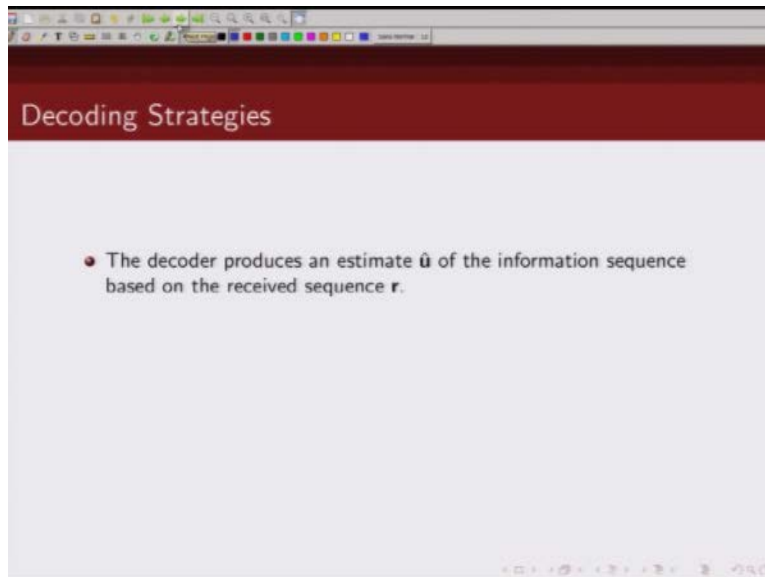
### Convolutional Codes

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

$$\begin{aligned} v_i^{(1)} &= u_i + u_{i-2} \\ v_i^{(2)} &= u_i + u_{i-1} + u_{i-2} \end{aligned}$$

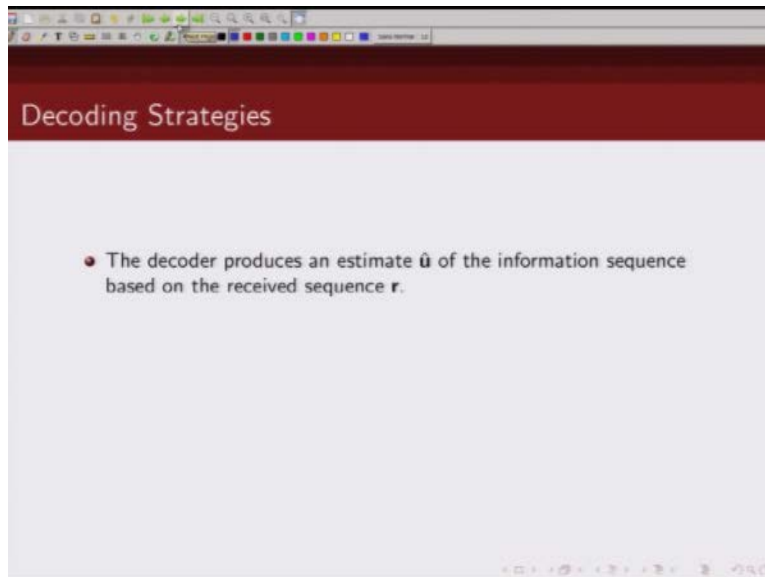
So you can see the difference here, the output not only depends on current input, but it also depends on past inputs similarly here, basically you can see. In a convolutional code the output depends on past inputs and outputs okay. So that is one of the major difference between convolutional codes and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

(Refer Slide Time: 14:46)



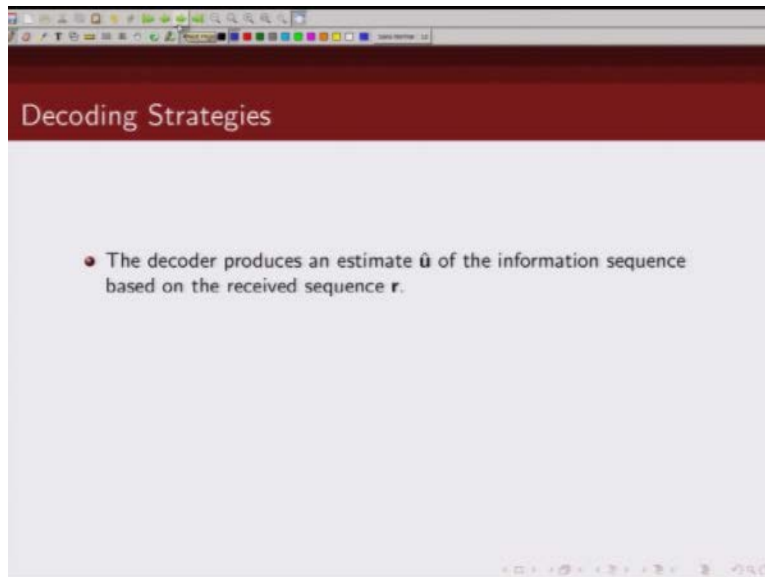
Now let us move to the topic of what sort of decoding strategy should we employ when we want to decode a code.

(Refer Slide Time: 15:02)



So now as I said a decoder objective is – it takes us input the demodulated signal  $R$ , and it has to produce an estimate of the information sequence  $\hat{u}$  all right.

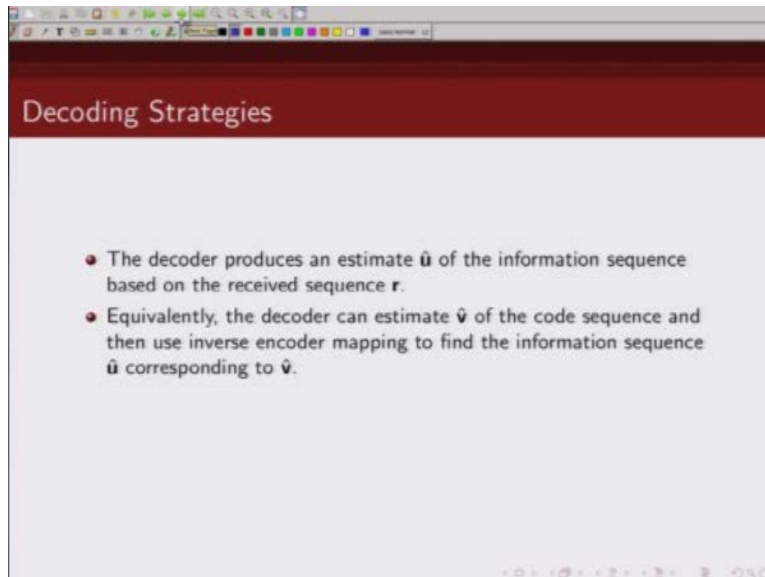
(Refer Slide Time: 15:22)



So the decoder produces an estimate of the information sequence based on what it has received of demodulated output which is  $r$ .



(Refer Slide Time: 15:34)

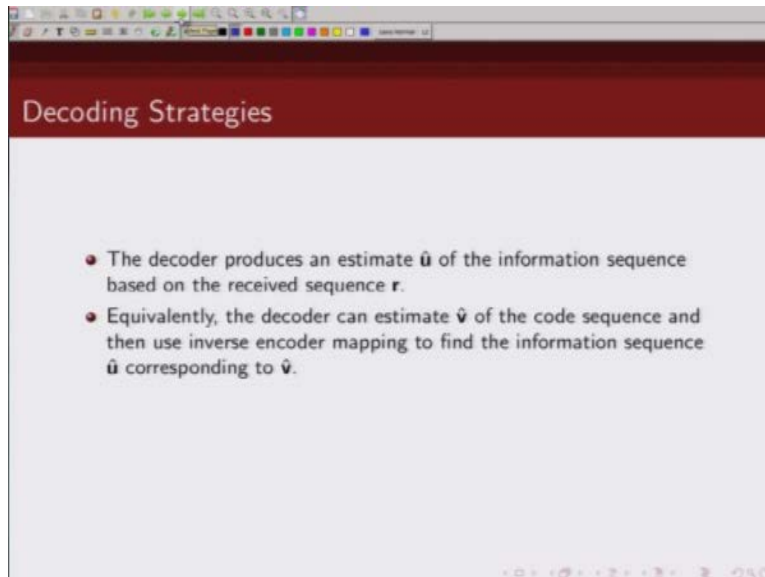


Decoding Strategies

- The decoder produces an estimate  $\hat{u}$  of the information sequence based on the received sequence  $r$ .
- Equivalently, the decoder can estimate  $\hat{v}$  of the code sequence and then use inverse encoder mapping to find the information sequence  $\hat{u}$  corresponding to  $\hat{v}$ .

Now we can say this estimation of the information sequence problem is equivalent to estimating the code sequence, because there is one to one mapping from a particular code word to the information sequence.

(Refer Slide Time: 15:56)

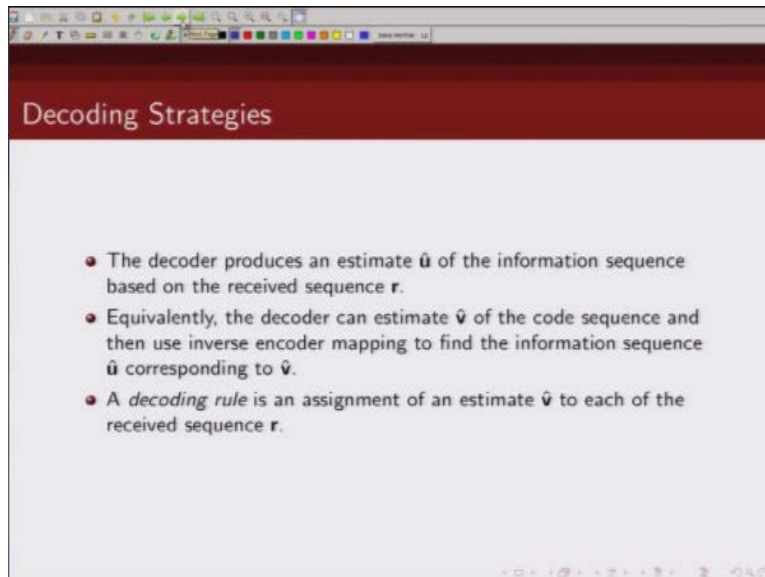


Decoding Strategies

- The decoder produces an estimate  $\hat{u}$  of the information sequence based on the received sequence  $r$ .
- Equivalently, the decoder can estimate  $\hat{v}$  of the code sequence and then use inverse encoder mapping to find the information sequence  $\hat{u}$  corresponding to  $\hat{v}$ .

So we can say equivalently the problem that decoder has to estimate is, it has to estimate the code sequence given a received sequence  $r$ , because there is one to one mapping from the message bits to the code bits.

(Refer Slide Time: 16:11)

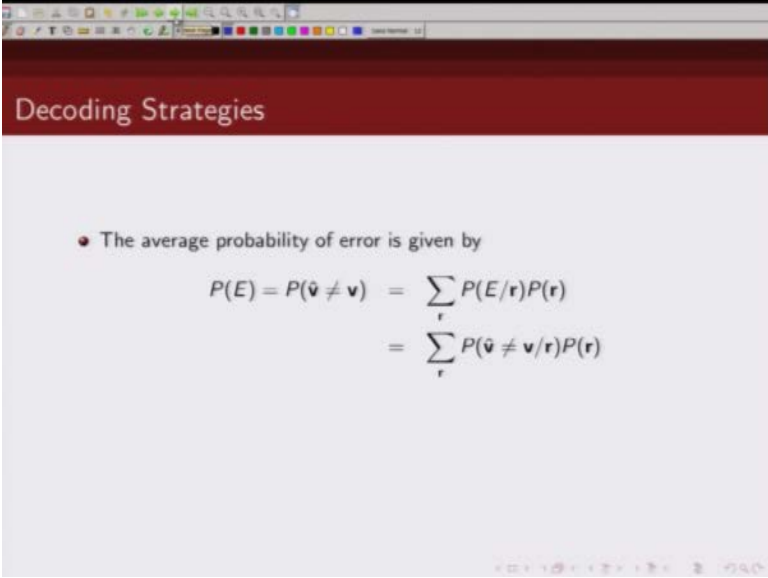


Decoding Strategies

- The decoder produces an estimate  $\hat{u}$  of the information sequence based on the received sequence  $r$ .
- Equivalently, the decoder can estimate  $\hat{v}$  of the code sequence and then use inverse encoder mapping to find the information sequence  $\hat{u}$  corresponding to  $\hat{v}$ .
- A *decoding rule* is an assignment of an estimate  $\hat{v}$  to each of the received sequence  $r$ .

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

(Refer Slide Time: 16:50)



Decoding Strategies

- The average probability of error is given by

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

Now one of the policies which we can use is basically to minimize probability of error. Now when does an error occur, when my decoded sequence is not same as my transmitter signal?

(Refer Slide Time: 17:05)

Decoding Strategies

- The average probability of error is given by

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

So my probability 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 some over all possible received sequence. An error is nothing but when  $\mathbf{v}$  is not same as  $\hat{\mathbf{v}}$ , so I can write this equation in this particular form.

(Refer Slide Time: 17:41)

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  $\mathbf{r}$ .

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  $P(\mathbf{r})$  and other is this term. Now whatever  $\hat{\mathbf{v}}$  I choose that does not change  $P(\mathbf{r})$ , so the choice of decoding rule does not change my  $P(\mathbf{r})$ . 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 okay. Now minimizing this term, minimizing this term is same as maximizing this term correct.

(Refer Slide Time: 18:50)

Decoding Strategies

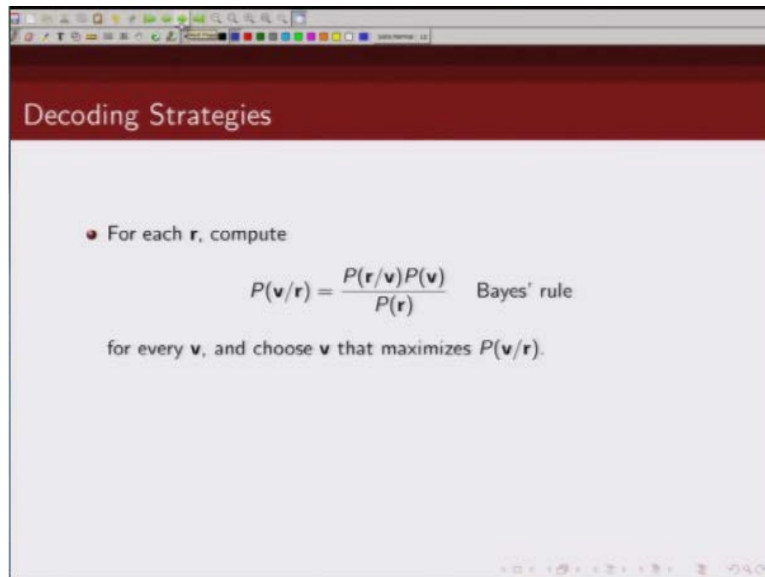
- The average probability of error is given by

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

- 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 the probability  $\hat{\mathbf{v}}$  is not same as  $(\mathbf{v}/\mathbf{r})$ , is equivalent to maximizing the probability that  $\hat{\mathbf{v}}$  is equal to  $(\mathbf{v}/\mathbf{r})$  okay.

(Refer Slide Time: 19:05)



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 have to maximize this, now using base rule we can write probability of (v/r) as probability of (r/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, so we can write our probability to maximize, so to maximize this, then becomes maximizing this quantity.



(Refer Slide Time: 19:51)

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  $P(\mathbf{v}/r)$ .

The slide features a dark red header with the title 'Decoding Strategies'. Below the header, a bullet point indicates the task: 'For each r, compute'. The central focus is the mathematical formula for Bayes' rule,  $P(\mathbf{v}/r) = \frac{P(r/\mathbf{v})P(\mathbf{v})}{P(r)}$ . Hand-drawn blue annotations highlight the components: a box around the numerator  $P(r/\mathbf{v})P(\mathbf{v})$  and a circle around the denominator  $P(r)$ . The text 'Bayes' rule' is placed to the right of the fraction. Below the formula, the instruction 'for every v, and choose v that maximizes P(v/r)' is written. The slide is presented in a window with a standard operating system taskbar at the top and a navigation bar at the bottom.

(Refer Slide Time: 19:52)

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

(Refer Slide Time: 20:20)

Decoding Strategies

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

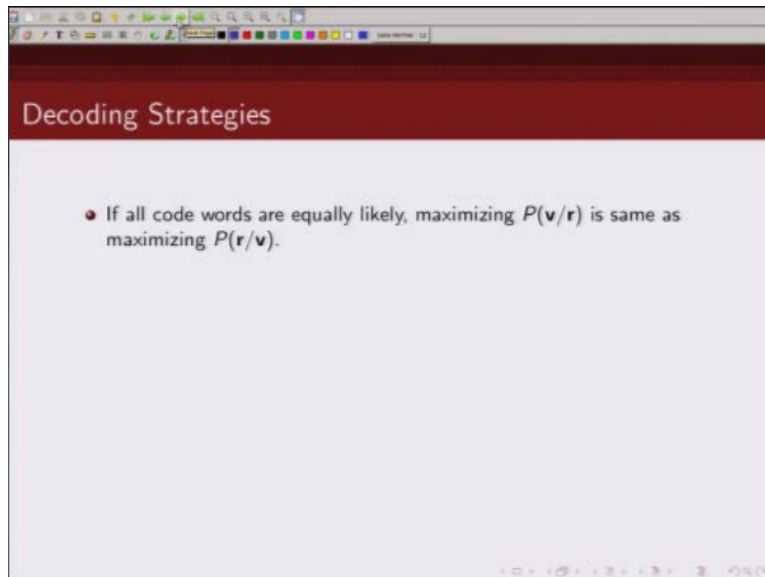
$$P(\mathbf{v}/\mathbf{r}) = \frac{P(\mathbf{r}/\mathbf{v})P(\mathbf{v})}{P(\mathbf{r})} \quad \text{Bayes' rule}$$

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  $P(\mathbf{v}/\mathbf{r})$  is maximized.

And a map decoder a maximum a-posteriori probability decoder is the one which will do exactly that, so it will choose a  $\hat{\mathbf{v}}$  such that this is – this probability is maximized.

(Refer Slide Time: 20:37)



Decoding Strategies

- If all code words are equally likely, maximizing  $P(\mathbf{v}/\mathbf{r})$  is same as maximizing  $P(\mathbf{r}/\mathbf{v})$ .

Now what happens if all code words are equally likely to happen, if all code words are equally likely to happen --

(Refer Slide Time: 20:44)

Decoding Strategies

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

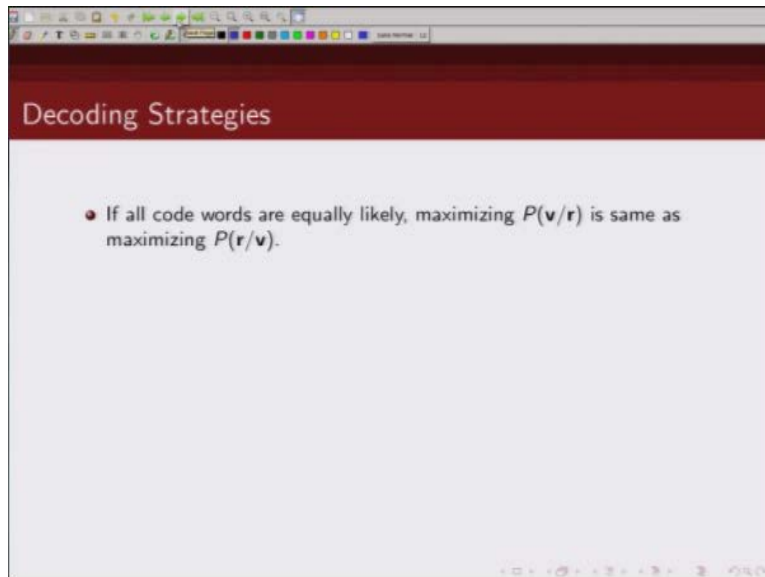
$$P(\mathbf{v}/\mathbf{r}) = \frac{P(\mathbf{r}/\mathbf{v})P(\mathbf{v})}{P(\mathbf{r})} \quad \text{Bayes' rule}$$

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  $P(\mathbf{v}/\mathbf{r})$  is maximized.

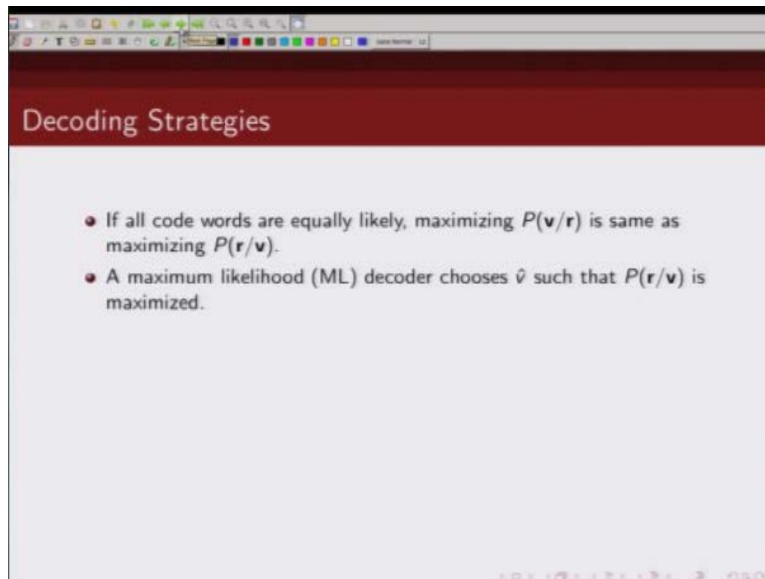
Then look at this term probability of  $\mathbf{v}$  will be same, so in that case maximizing  $P(\mathbf{v}/\mathbf{r})$  is same as maximizing  $P(\mathbf{r}/\mathbf{v})$ .

(Refer Slide Time: 21:00)



So that is what we are seeing, if all code words are equally likely, then maximizing  $P(\mathbf{v}/\mathbf{r})$  is same as maximizing this likelihood ratio, likelihood function  $P(\mathbf{r}/\mathbf{v})$ .

(Refer Slide Time: 21:18)



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.

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

(Refer Slide Time: 21:29)

Decoding Strategies

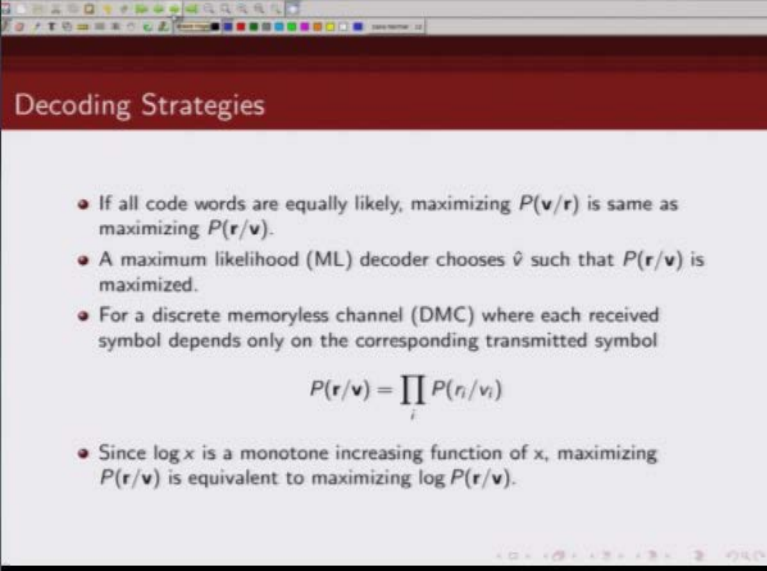
- If all code words are equally likely, maximizing  $P(\mathbf{v}/\mathbf{r})$  is same as maximizing  $P(\mathbf{r}/\mathbf{v})$ .
- A maximum likelihood (ML) decoder chooses  $\hat{v}$  such that  $P(\mathbf{r}/\mathbf{v})$  is maximized.
- For a discrete memoryless channel (DMC) where each received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

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

Now if we go see that our channel is discrete memory less channel, in other words we can write the probability for a discrete memory less channel we can write probability of received sequence  $\mathbf{r}$ , given transmit sequence  $\mathbf{v}$  we can write it as product of these individual probabilities.



(Refer Slide Time: 21:52)



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  $\log P(\mathbf{r}/\mathbf{v})$ .

So if that happens then we can further simplify our maximizing criteria, so we want to maximize this is same as maximizing this. Now since  $\log x$  is a monotone increasing function of  $x$ , we can say maximizing this probability is same, is equivalent to maximizing  $\log P(\mathbf{r}/\mathbf{v})$ . Now if we do that then this product becomes  $\sum$  okay. So then we can basically write this as basically then  $\log$  of  $P(\mathbf{r}/\mathbf{v})$  will become basically  $\sum$  and this will be basically of course there will be some log term here.

(Refer Slide Time: 22:53)

**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

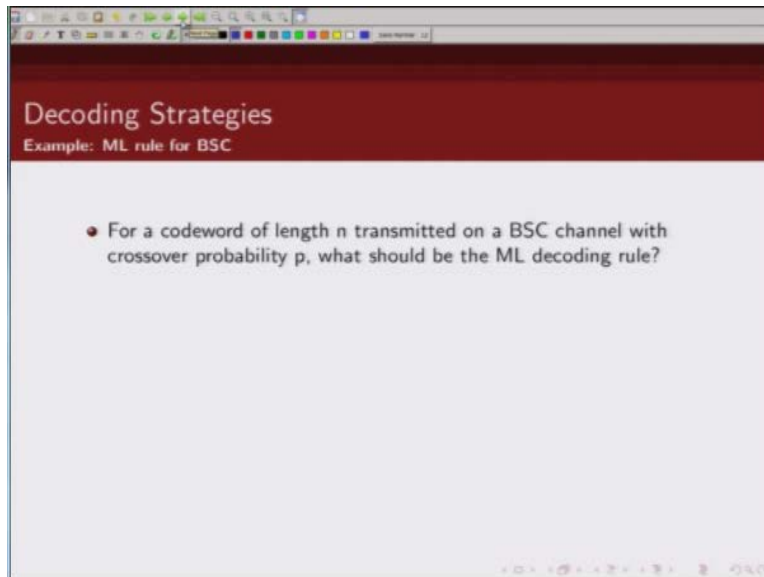
$$P(\mathbf{r}/\mathbf{v}) = \prod_i P(r_i/v_i)$$

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

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

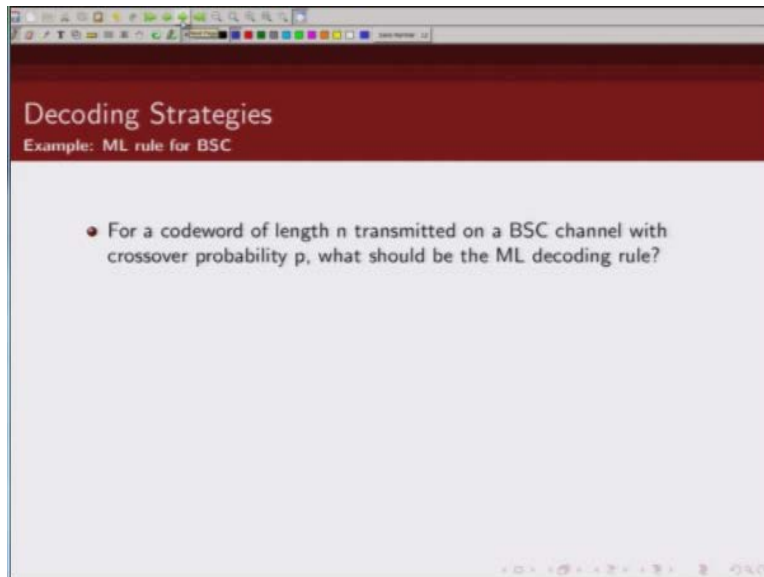
And this is basically much easier to compute okay.

(Refer Slide Time: 22:58)



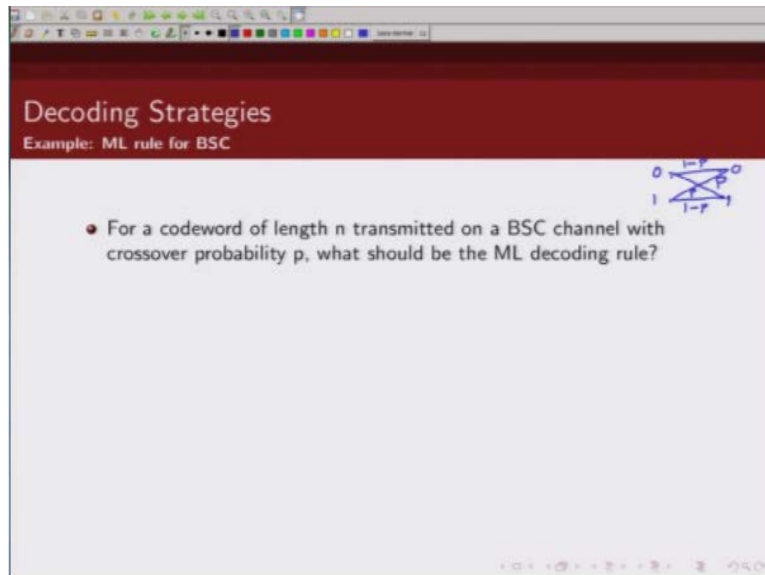
So let us take an example we are interested in finding what would be the maximum likelihood decoding rule for a binary symmetric channel. Now recall what is a binary symmetric channel.

(Refer Slide Time: 23:15)



There are two inputs this is 011, two outputs 011 with probability  $1-P$ , I received my bits correctly and there is a crossover probability of  $P$  okay.

(Refer Slide Time: 23:33)



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?

The diagram shows a channel with two input nodes labeled 0 and 1, and two output nodes labeled 0 and 1. The transition probabilities are:  $P(0 \rightarrow 0) = 1-p$ ,  $P(0 \rightarrow 1) = p$ ,  $P(1 \rightarrow 0) = p$ , and  $P(1 \rightarrow 1) = 1-p$ .

So the question I am asking is, if I have a code word of length  $n$ , which is transmitted over a binary symmetric channel whose crossover probability is  $p$ , what should be my maximum likelihood decoding rule? 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  $(r/v)$  which is equivalent to maximizing  $\log P(r/v)$ . So let us try to compute what is the  $\log (r/v)$  okay.

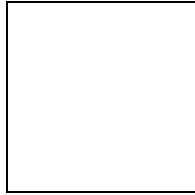
(Refer Slide Time: 24:27)

Now before I calculate  $P(r/v)$  let me introduce another term which is called hamming distance. Now what is hamming distance between two code words, so the hamming distance between two code words are to interpoles let us call it hamming distance between  $r$  and  $v$ , both are  $n$ -bit vector, basically.

So the hamming distance between  $r$  and  $v$  is defined as number of positions in which  $r$  and  $v$  are differing. So for example, if let us say  $r = 111011$  and  $v = 011101$  then what is the hamming distance? It is differing in the first location 1, is not differing here, not differing here, is differing here that is two, is differing in this location that is three, is not differing in this location.

So  $r$  and  $v$  differs in three location, so one is this location, other is this location, and third is this location.

(Refer Slide Time: 25:38)



So the hamming distance between  $r$  and  $v$  is three in this case okay. Now when we are sending an  $n$ -bit code word or a binary symmetric channel what happens, some of the bits will get flipped with probability crossover probability  $P$ . Let us denote those numbers of flip bits by  $d$ .

(Refer Slide Time: 26:05)

**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. And when  $\mathbf{r}$  and  $\mathbf{v}$  are not same that means those are the locations where error has occurred. So number of positions that got flipped as a result of sending this code word or binary symmetric channel that is denoted by this  $d(\mathbf{r}, \mathbf{v})$  and the remaining number of bits which did not get changed that is basically  $n-d(\mathbf{r}, \mathbf{v})$ . So these many bits did not get changed and these many bits got flipped.



(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 diagrams and examples on the slide:

- A diagram showing a square with nodes 0 and 1 at the corners. Transitions are labeled with  $1-p$  and  $p$ .
- Handwritten binary strings:  $\mathbf{r} = 111011$  and  $\mathbf{v} = 011101$ . The Hamming distance  $d(\mathbf{r}, \mathbf{v}) = 4$  is indicated by underlining the differing bits.
- Handwritten calculation:  $\frac{n-d(\mathbf{r}, \mathbf{v})}{d(\mathbf{r}, \mathbf{v})}$ .

So what is the probability that  $d$ -bits got flipped that is given by  $p^{d(\mathbf{r}, \mathbf{v})}$  and what is the probability, that  $n-d$  bits were received correctly that is given by  $1-p^{(n-d(\mathbf{r}, \mathbf{v}))}$  this quantity.

(Refer Slide Time: 27:18)

**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: A square with nodes 0 and 1 at the corners. The top edge is labeled  $1-p$ , the bottom edge is labeled  $1-p$ , the left edge is labeled  $p$ , and the right edge is labeled  $p$ .

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

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

Handwritten simplification:  $(1-p)^{\frac{(n-d(r,v))}{p^{d(r,v)}}$

So we can write probability of  $(\mathbf{r}, \mathbf{v})$  as  $p^{d(1-p)^{n-d}}$ . Now 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  $d(\mathbf{r}, \mathbf{v})$  out, so what we will get is  $d(\mathbf{r}, \mathbf{v}) \log p/1-p + n \log(1-p)$ . So to maximize this probability we have to choose our  $\hat{\mathbf{v}}$  such that this is maximized. Now look closely at both of these terms, let us first look at this term thus this term depends on selection of  $\mathbf{v}$ ? No.

It depends on  $n$  which is code word length it depends on crossover probability  $p$ . So whatever way we chose 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 half, if that happens.

(Refer Slide Time: 29:06)

**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  $r_i \neq v_i$ .

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

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

What happens to this ratio,  $p/1-p$  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  $-d(\mathbf{r}, \mathbf{v})$  correct? So a maximum likelihood decoder will choose a  $\mathbf{v}$  such that  $-d(\mathbf{r}, \mathbf{v})$  is maximized. In other words we should choose a code word  $\mathbf{v}$  in such a way, such that  $d(\mathbf{r}, \mathbf{v})$  is minimized. When  $d(\mathbf{r}, \mathbf{v})$  is minimized then only  $-d(\mathbf{r}, \mathbf{v})$  will be maximized.

(Refer Slide Time: 30:05)

Decoding Strategies  
Example: ML rule for BSC

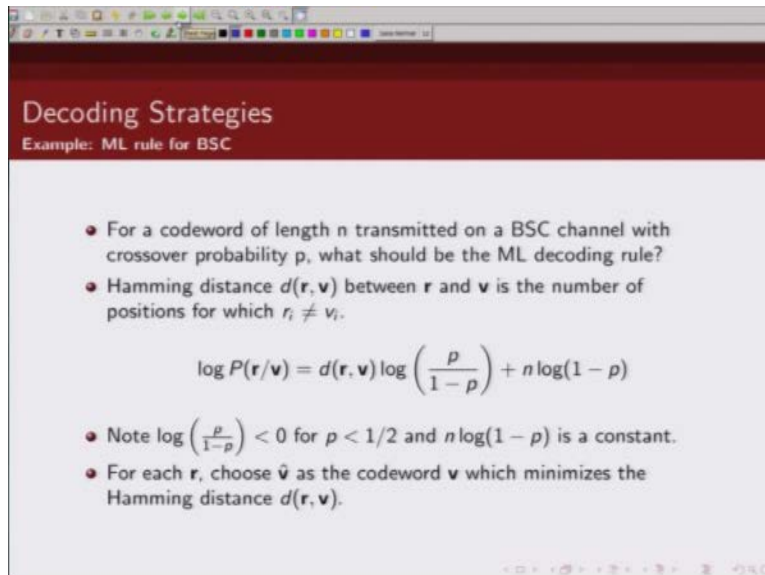
- For a codeword of length  $n$  transmitted on a BSC channel with crossover probability  $p$ , what should be the ML decoding rule?
- Hamming distance  $d(\mathbf{r}, \mathbf{v})$  between  $\mathbf{r}$  and  $\mathbf{v}$  is the number of positions for which  $r_i \neq v_i$ .

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

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

So that is what we are seeing here,  $\log(P/1-P)$  is less than 0, so this will be a negative quantity. When you want to do 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: 30:26)



Decoding Strategies

Example: ML rule for BSC

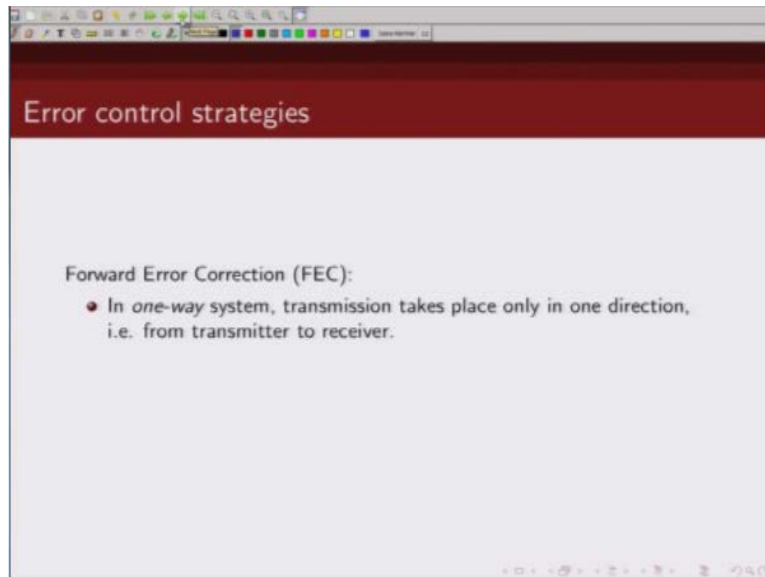
- For a codeword of length  $n$  transmitted on a BSC channel with crossover probability  $p$ , what should be the ML decoding rule?
- Hamming distance  $d(\mathbf{r}, \mathbf{v})$  between  $\mathbf{r}$  and  $\mathbf{v}$  is the number of positions for which  $r_i \neq v_i$ .

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

- 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 code word  $\mathbf{v}$  such that hamming distance between the code word  $\mathbf{v}$  and the received sequence is minimized and that makes sense. And that is our maximum likelihood decoding rule.

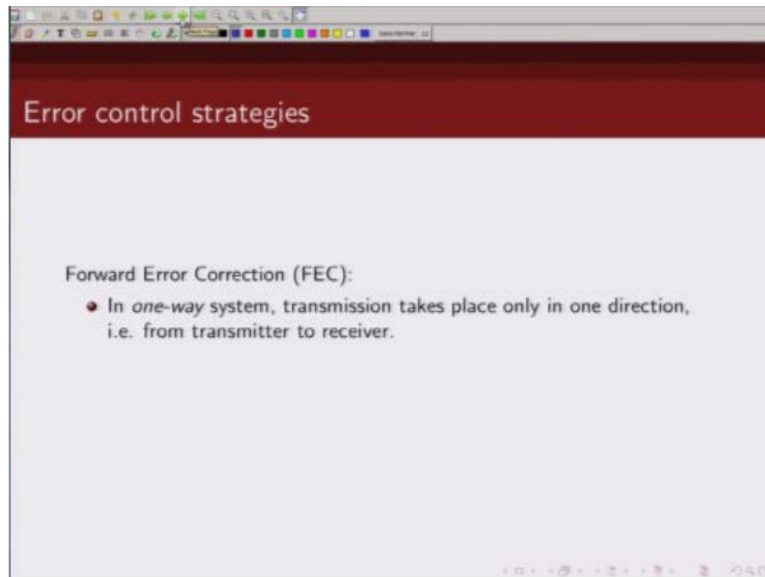
(Refer Slide Time: 30:59)



The image shows a screenshot of a presentation slide. The slide has a dark red header with the text "Error control strategies" in white. Below the header, the text "Forward Error Correction (FEC):" is displayed. Underneath this, there is a single bullet point: "• In *one-way* system, transmission takes place only in one direction, i.e. from transmitter to receiver." The slide is presented in a window with a standard operating system taskbar at the top and a navigation bar at the bottom.

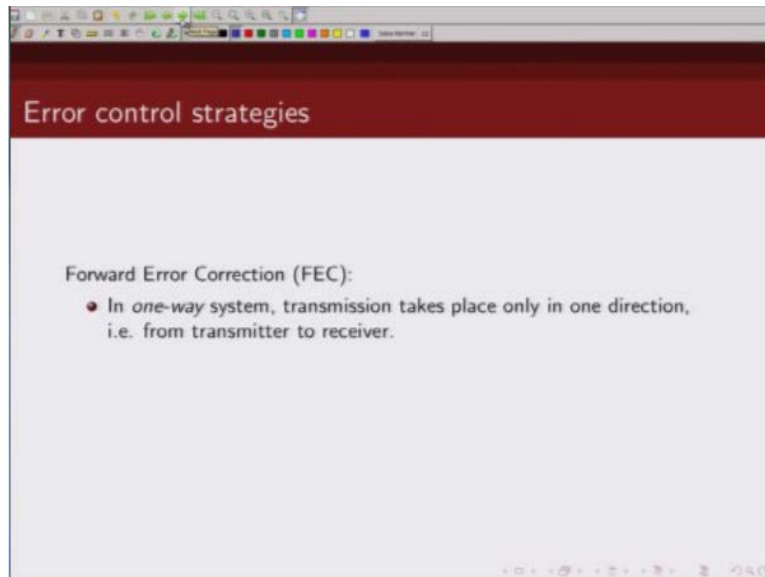
So finally I am going to conclude this lecture with definition of few error correcting strategies.

(Refer Slide Time: 31:12)



The first one which I am going to describe is what is known as FEC Forward Error Correction. So in systems where there is no feedback from the receiver to the transmitter we are calling those systems as one way system.

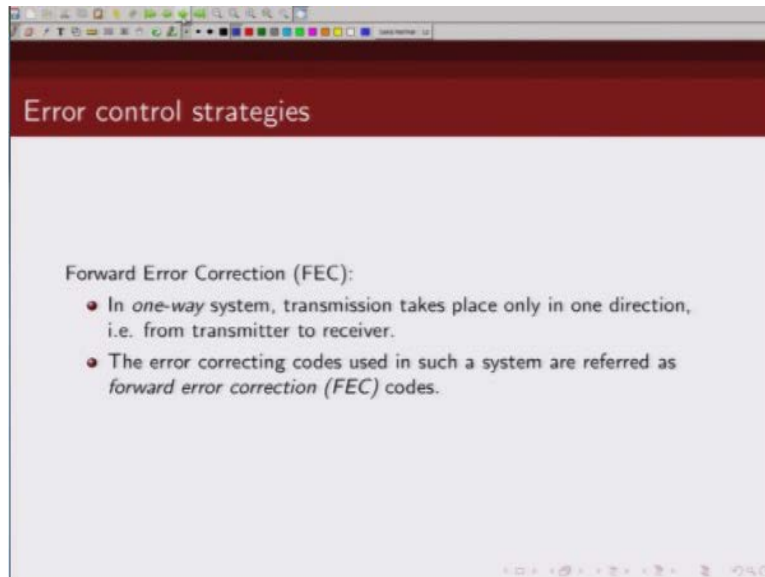
(Refer Slide Time: 31:30)



Where transmission happens only in one direction from transmitter to receiver.



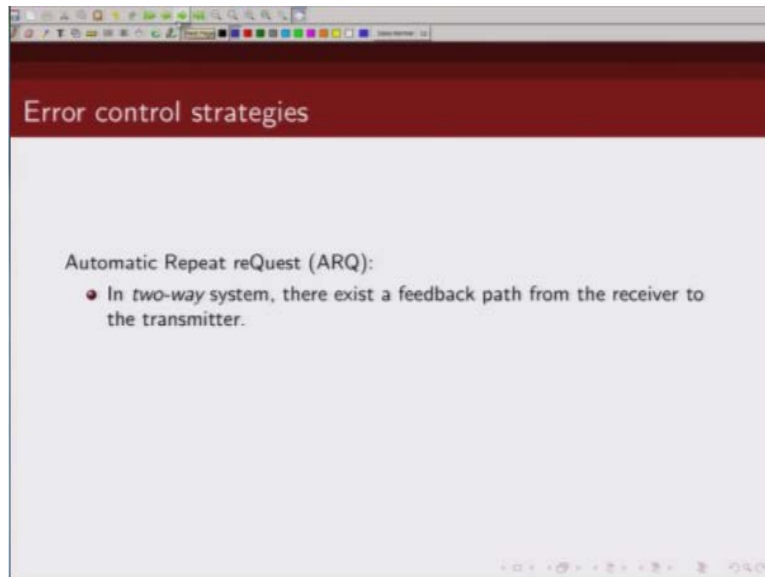
(Refer Slide Time: 31:35)



The image shows a screenshot of a presentation slide. The slide has a dark red header with the text "Error control strategies" in white. Below the header, the text "Forward Error Correction (FEC):" is followed by two bullet points. The first bullet point states: "In *one-way* system, transmission takes place only in one direction, i.e. from transmitter to receiver." The second bullet point states: "The error correcting codes used in such a system are referred as *forward error correction (FEC) codes*." The slide is displayed in a window with a standard operating system taskbar at the top and a navigation bar at the bottom.

In those systems the error correcting codes that are used are known as FEC. So when you hear this term FEC code is basically means basically when we are sending – so this is the 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: 32:10)



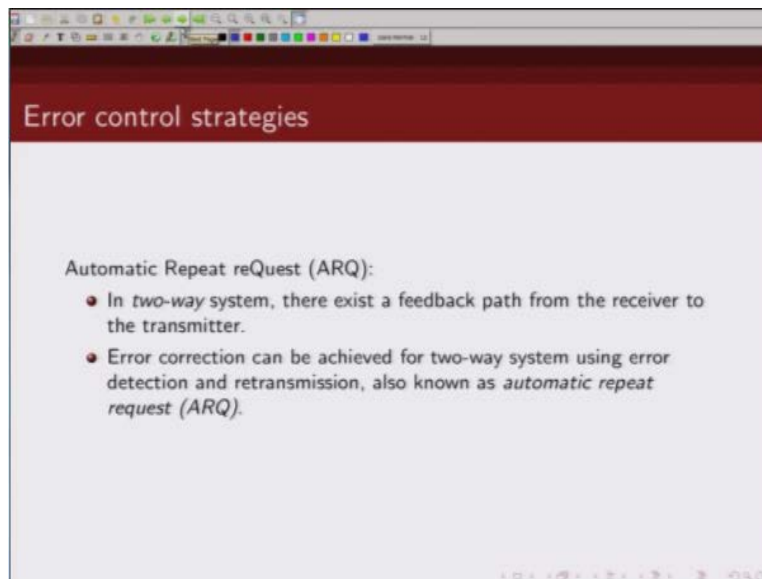
Error control strategies

Automatic Repeat reQuest (ARQ):

- In two-way system, there exist a feedback path from the receiver to the transmitter.

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

(Refer Slide Time: 32:20)

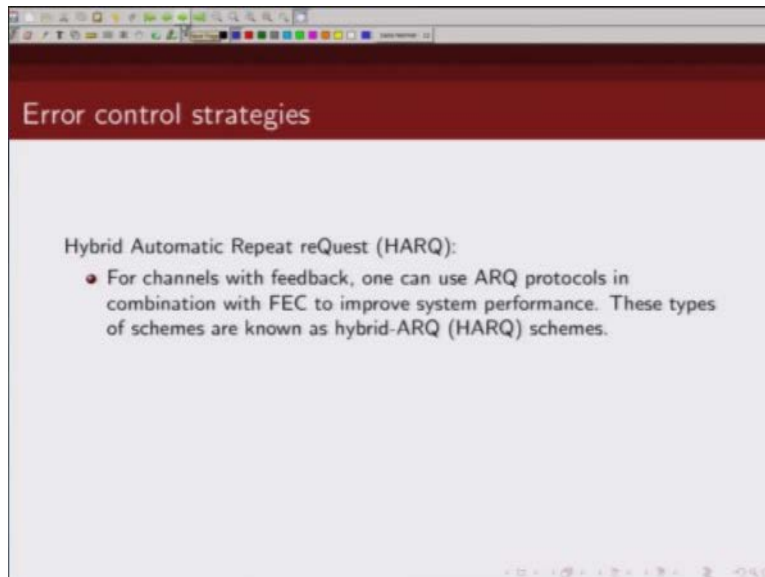


So for these systems basically the error correcting strategy which is used is what is known as automatic repeat request. Now how does this works, so you initially send some coded packets where you just use parity bits for error detection. So you send your information bits and some few parity bits for 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 is 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 bit for the error correction.

So this is typically useful if that links are very good, you just are sending uncoded packet with some bits for error detection and occasionally when the packets are not received correctly then you ask for retransmission.

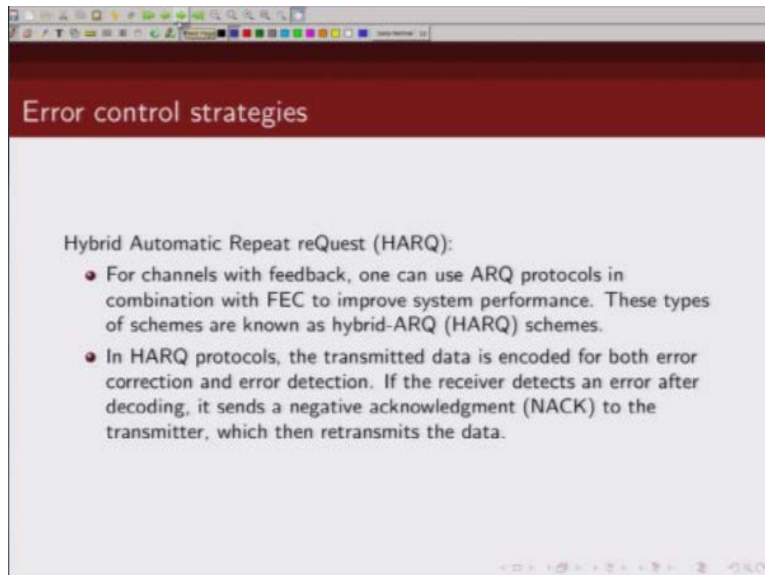
(Refer Slide Time: 33:57)



A strategy that combines both forward error correction and ARQ is known as hybrid ARQ system. 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 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 ARQ is the combination of forward error correcting scheme and automatic repeat request scheme.

(Refer Slide Time: 34:42)



Typically it is in a communication system you will see a combination of both forward error correcting schemes and hybrid ARQ scheme used. So with this I am going to conclude this lecture. Thank you.

### **Acknowledgment**

**Ministry of Human Resources & Development**

**Prof. Satyaki Roy**

**Co-ordinator, NPTEL IIT Kanpur**

**NPTEL Team**

**Sanjay Pal**

**Ashish Singh**

**Badal Pradhan**

**Tapobrata Das**

**Ram Chandra**

**Dilip Tripathi**

**Manoj Shrivastava**

**Padam Shukla**

**Sanjay Mishra**

**Shubham Rawat**

**Shikha Gupta**

**K. K. Mishra**

**Aradhana Singh**

**Sweta**

**Ashutosh Gairola**

**Dilip Katiyar**

**Sharwan**

**Hari Ram**

**Bhadra Rao**

**Puneet Kumar Bajpai**

**Lalty Dutta**

**Ajay Kanaujia**

**Shivendra Kumar Tiwari**

**an IIT Kanpur Production**

**©copyright reserved**