

# Digital Communication using GNU Radio

Prof. Kumar Appiah

Department of Electrical Engineering

Indian Institute of Technology Bombay

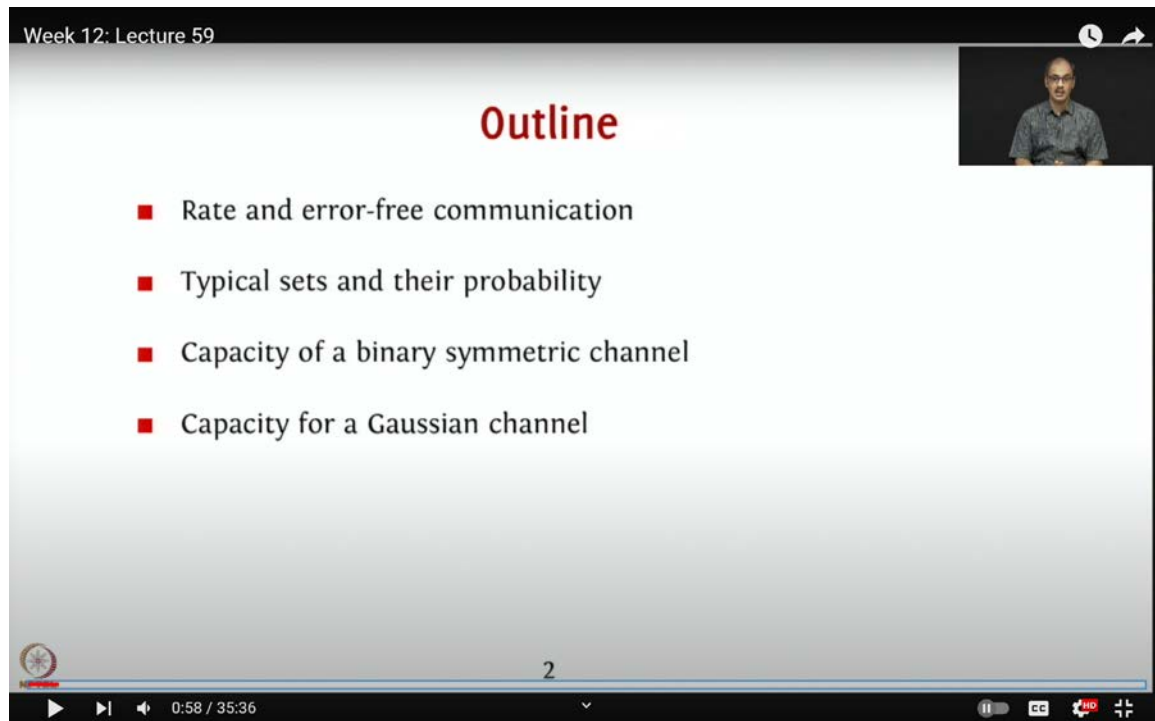
Week-12

Lecture-59

## Rate and Error-Free Communication

Welcome to this lecture on Digital Communication using GNU Radio. My name is Kumar Appiah. In this session, we will explore the concept of channel capacity, which relates to the maximum rate at which error-free communication can be achieved over a given channel. Although our primary focus will be on the binary symmetric channel, we will also touch on how these concepts extend to other types of channels.

(Refer Slide Time: 00:58)



The screenshot shows a video player interface for a lecture. The title bar at the top left reads "Week 12: Lecture 59". The main content area has a red heading "Outline" followed by a bulleted list of four topics: "Rate and error-free communication", "Typical sets and their probability", "Capacity of a binary symmetric channel", and "Capacity for a Gaussian channel". A small video inset in the top right corner shows the professor. The bottom of the slide features a navigation bar with a play button, a progress bar at 0:58 / 35:36, and various control icons.

Week 12: Lecture 59

### Outline

- Rate and error-free communication
- Typical sets and their probability
- Capacity of a binary symmetric channel
- Capacity for a Gaussian channel

2

0:58 / 35:36

Here's the outline of our discussion:

1. Rate and Error-Free Communication: We will start by discussing the rate of error-free

communication.

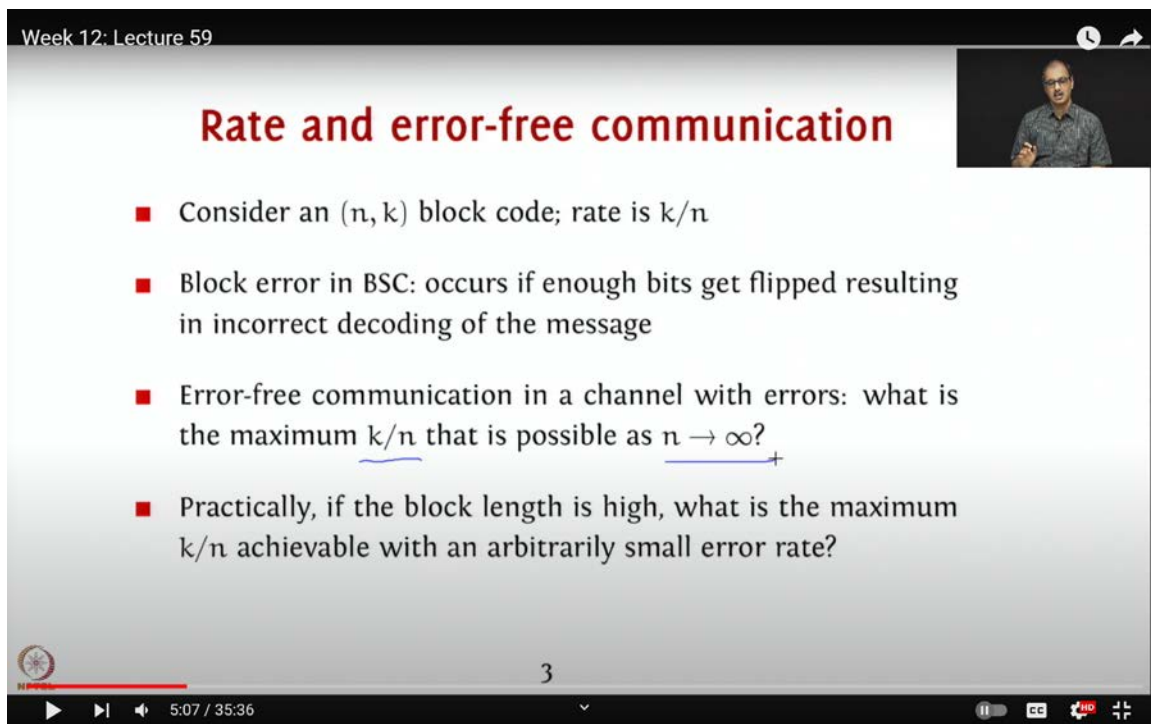
2. Typical Sets and Their Probability: We will briefly cover typical sets and their associated probabilities.

3. Binary Symmetric Channel Capacity: We will specify the capacity of a binary symmetric channel.

4. Gaussian Channel Capacity: We will also touch on the capacity of a Gaussian channel, which is more practical and relevant to the simulations you have encountered.

5. Implications of Capacity: Finally, we will summarize what channel capacity means for practical communication systems.

(Refer Slide Time: 05:07)



The screenshot shows a video player interface for a lecture. The title bar at the top left says 'Week 12: Lecture 59'. The main slide content is titled 'Rate and error-free communication' in red. It contains four bullet points: 1. Consider an  $(n, k)$  block code; rate is  $k/n$ . 2. Block error in BSC: occurs if enough bits get flipped resulting in incorrect decoding of the message. 3. Error-free communication in a channel with errors: what is the maximum  $k/n$  that is possible as  $n \rightarrow \infty$ ? (with underlines and arrows indicating the limit process). 4. Practically, if the block length is high, what is the maximum  $k/n$  achievable with an arbitrarily small error rate? A small video inset in the top right shows a man speaking. The video player controls at the bottom show a progress bar at 5:07 / 35:36 and a slide number '3'.

Week 12: Lecture 59

## Rate and error-free communication

- Consider an  $(n, k)$  block code; rate is  $k/n$
- Block error in BSC: occurs if enough bits get flipped resulting in incorrect decoding of the message
- Error-free communication in a channel with errors: what is the maximum  $k/n$  that is possible as  $n \rightarrow \infty$ ?
- Practically, if the block length is high, what is the maximum  $k/n$  achievable with an arbitrarily small error rate?

3

5:07 / 35:36

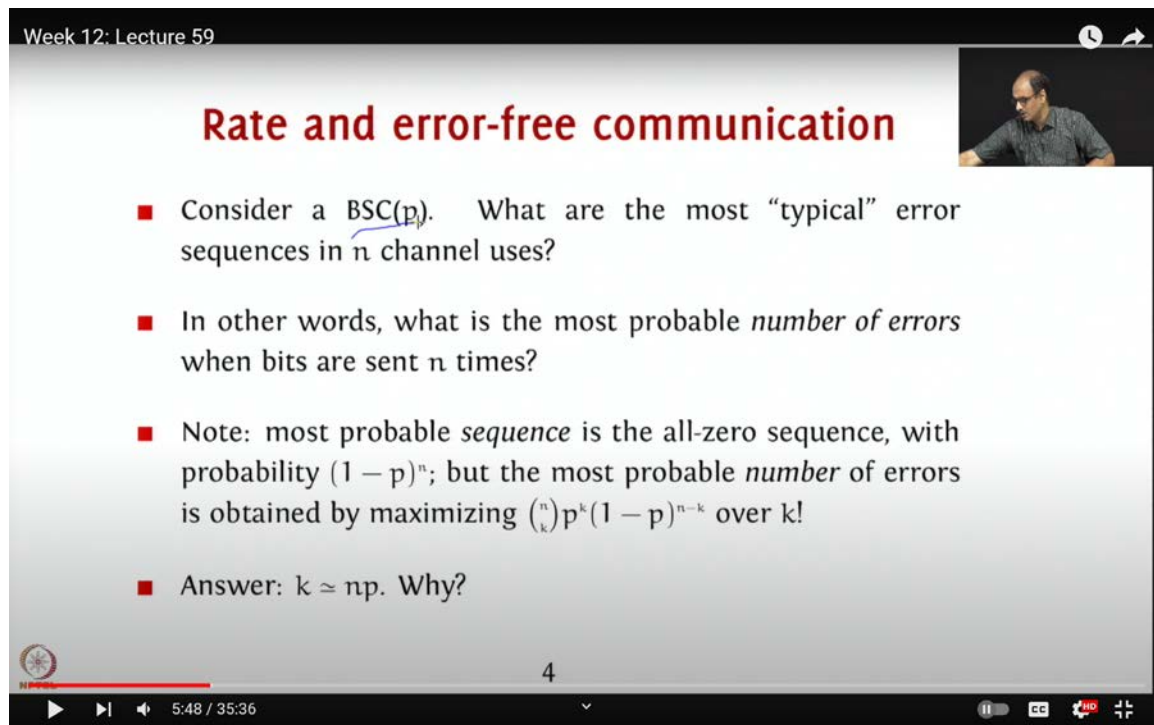
To begin with, let's examine the rate of error-free communication. In the context of an  $(n, k)$  block code, as discussed in previous lectures, the rate is given by  $\frac{k}{n}$ . The block error in a binary symmetric channel occurs when a sufficient number of bits are flipped. In such a

noisy channel, if no bits are flipped, there is no issue. However, if bit flips do occur, they can potentially be corrected.

For instance, we've seen error correction mechanisms like the Hamming code and repetition code. These codes are designed to be robust against bit flips, enabling the receiver to correctly interpret the transmitted message even if errors occur.

Now, what does error-free communication entail? Hypothetically, error-free communication means having a channel that introduces errors but incorporating enough redundancy so that all these errors can be managed effectively.

(Refer Slide Time: 05:48)



Week 12: Lecture 59

## Rate and error-free communication

- Consider a  $BSC(p)$ . What are the most “typical” error sequences in  $n$  channel uses?
- In other words, what is the most probable *number of errors* when bits are sent  $n$  times?
- Note: most probable *sequence* is the all-zero sequence, with probability  $(1 - p)^n$ ; but the most probable *number of errors* is obtained by maximizing  $\binom{n}{k} p^k (1 - p)^{n-k}$  over  $k$ !
- Answer:  $k \simeq np$ . Why?

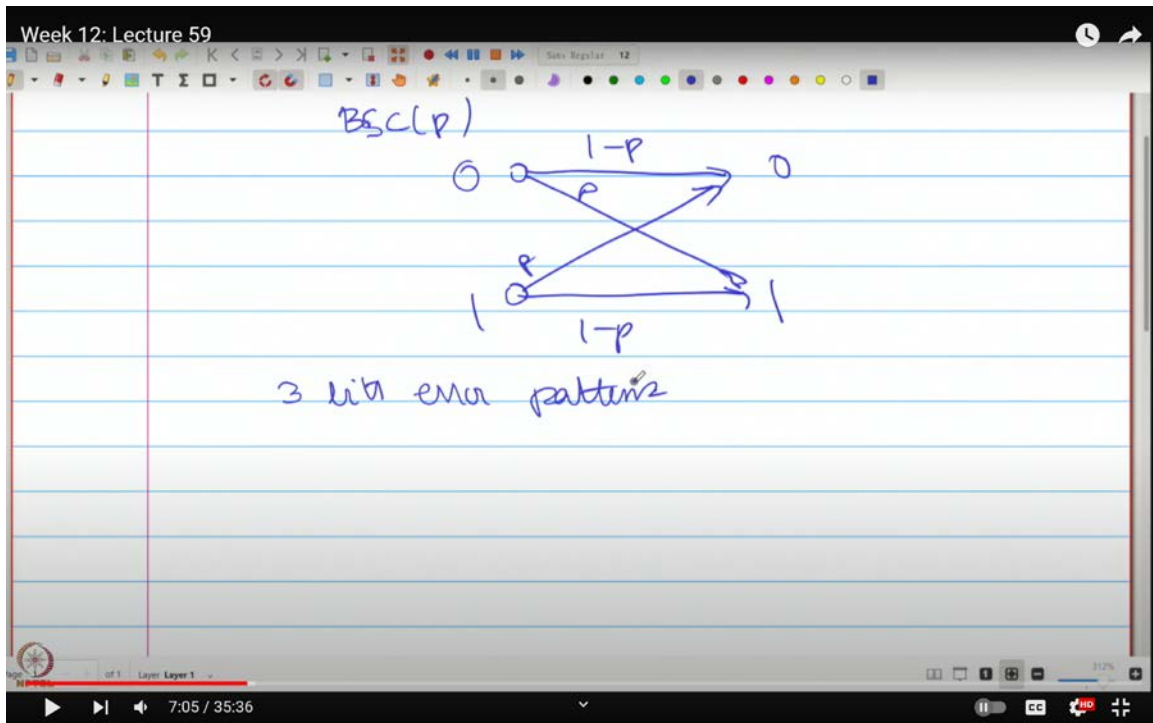
4

5:48 / 35:36

Let's consider an example. Suppose we have a binary symmetric channel with a bit-flip probability  $p = 0.2$ . This means there is a 20% chance that each bit will be flipped. However, if we employ a repetition code and repeat each bit 11 times, we can significantly reduce the effective error rate. Even though the channel flips bits, the repetition code can handle these errors and recover the original data. The trade-off here is that the rate of the code becomes quite poor. For an 11-repetition code, the rate is  $\frac{1}{11}$ .

The key question we are addressing is: What is the maximum achievable rate  $\frac{k}{n}$  as  $n$  approaches infinity? Essentially, we are interested in finding the maximum possible rate for a very large block length. When you allow a large number of bits to be sent through the channel, patterns in the types of errors introduced by the channel emerge. Understanding these patterns enables the design of more effective codes that can counteract these errors. This process helps us approach the channel's capacity.

(Refer Slide Time: 07:05)



In practice, with a very high block length, what is the maximum  $\frac{k}{n}$  that can be achieved with an arbitrarily small error rate? As students of probability, you know that with any non-zero error probability in a binary symmetric channel, and for any finite  $n$ , there will always be some situations where a large number of errors occur, regardless of the code used. Hence, decoding errors will always be present.

Let's consider a practical scenario. Suppose you require the error rate to be less than  $10^{-9}$  or even  $10^{-12}$ . In such cases, you might design a code to meet these stringent requirements. The simplest approach might be to use a repetition code, but this is inefficient because

achieving a very low error probability requires a significantly large  $n$ , making the rate  $\frac{1}{n}$  approach zero as  $n$  grows.

(Refer Slide Time: 10:18)

The image shows a digital screen with handwritten notes in blue and red ink. The title at the top is "Week 12: Lecture 59". The notes are organized as follows:

- At the top, there is a small diagram of a bit being flipped with probability  $p$  and staying the same with probability  $1-p$ .
- The main heading is "3 bit error patterns".
- Below this, a list of 8 3-bit patterns is shown, grouped by the number of errors:
  - 0 errors: 000, 111
  - 1 error: 001, 010, 100, 011, 101, 110
  - 2 errors: 100, 011, 101, 110
  - 3 errors: 111
- Next to the patterns, the probabilities are calculated:
  - 0 errors:  $(1-p)^3$  (for 000) and  $(1-p)^3$  (for 111)
  - 1 error:  $p(1-p)^2$  (for each of the 6 patterns), totaling  $3p(1-p)^2$
  - 2 errors:  $p^2(1-p)$  (for each of the 6 patterns), totaling  $3p^2(1-p)$
  - 3 errors:  $p^3$  (for 111)
- At the bottom, the heading "n-bit error pattern" is written.

The video player interface at the bottom shows the time 10:18 / 35:36.

Therefore, we aim to find the highest possible  $\frac{k}{n}$  that allows for error-free communication over a channel with errors. Now, turning our attention back to a binary symmetric channel with an error probability  $p$ , we need to identify the most typical error sequences over  $n$  channel uses. Here, "typical" refers to sequences that are probable, meaning they are not the rarest but rather the most likely error patterns. For example, if  $p < 0.5$ , the most probable error sequence in a binary symmetric channel is one where no errors occur at all. Our goal is to determine the number of errors that are most likely to occur when bits are transmitted  $n$  times.

Let's delve into this in greater detail. Consider a binary symmetric channel with an error probability  $p$ . In this channel, a zero is transmitted as zero with a probability of  $1 - p$ , and a one is transmitted as one with a probability of  $1 - p$ . Conversely, each bit has a probability  $p$  of being flipped.

(Refer Slide Time: 13:09)

Week 12: Lecture 59

000  
011  
101  
110  
111

2 errors each  $\rightarrow p^2(1-p)$

3 errors  $\rightarrow p^3$

$n$ -bit error pattern

0 errors  $\rightarrow (1-p)^n$

1 error  $\rightarrow np(1-p)^{n-1} \binom{n}{1} p(1-p)^{n-1}$

$l$  errors  $\rightarrow \binom{n}{l} p^l (1-p)^{n-l}$

Now, let's examine the behavior of this channel over multiple transmissions, specifically focusing on 3 bits. We will analyze the error patterns based on their Hamming weight, which represents the number of bit errors:

- 0 errors: 000
- 1 error: 001, 010, 100
- 2 errors: 011, 101, 110
- 3 errors: 111

Grouping these patterns:

- 0 errors: Probability is  $(1 - p)^3$
- 1 error: Probability is  $3p(1 - p)^2$
- 2 errors: Probability is  $3p^2(1 - p)$
- 3 errors: Probability is  $p^3$

When  $p$  is less than 0.5, the probability of having zero errors (000) is the highest among

these patterns. For example, if  $p$  is 0.1, then:

- The probability of 0 errors is approximately  $(0.9)^3$
- The probability of 1 error is approximately  $3 \cdot 0.1 \cdot (0.9)^2$
- The probability of 2 errors is approximately  $3 \cdot (0.1)^2 \cdot 0.9$
- The probability of 3 errors is  $(0.1)^3$

(Refer Slide Time: 15:08)

Week 12: Lecture 59

$n$  bit error pattern

0 errors  $\rightarrow (1-p)^n$

1 error  $\rightarrow np(1-p)^{n-1} \quad \binom{n}{1} p (1-p)^{n-1}$

$l$  errors  $\rightarrow \binom{n}{l} p^l (1-p)^{n-l}$

Aside: biased coin  $IP(H) = (1-p)$

$IP(HHHH \dots H) = (1-p)^n$

$IP(l \text{ tails, } n-l \text{ heads}) = \binom{n}{l} p^l (1-p)^{n-l}$

Summing these probabilities should add up to 1, which confirms that all possible error patterns are accounted for. For instance, with  $p = 0.1$ :

- The probability of having 0 errors is around  $0.9^3$
- The probability of having 1 error is  $3 \cdot 0.1 \cdot 0.9^2$
- The probability of having 2 errors is  $3 \cdot 0.1^2 \cdot 0.9$
- The probability of having 3 errors is  $0.1^3$

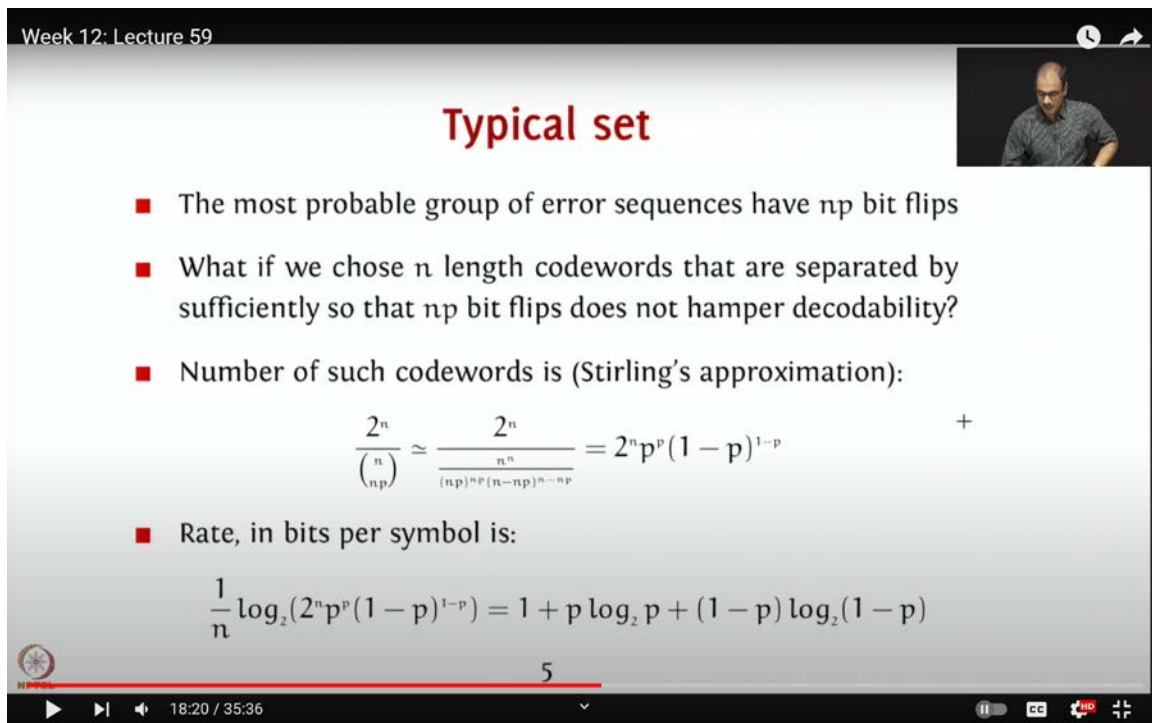
Now, let's generalize to  $n$  bit patterns. Instead of targeting individual error patterns, it's more practical to focus on the number of errors the channel introduces across multiple uses.



We calculate the cumulative probability for all possible sequences to understand the overall error distribution.

In this context, we are investigating the occurrence of typical sequences. To clarify, the most typical sequences are those where the probability remains consistent, even with larger block lengths. For instance, consider the probability expressions  $(1 - p)^n$  and  $n \cdot p \cdot (1 - p)^{n-1}$ . While  $(1 - p)^n$  can be a large number,  $n \cdot p \cdot (1 - p)^{n-1}$  might vary.

(Refer Slide Time: 18:20)



Week 12: Lecture 59

## Typical set

- The most probable group of error sequences have  $np$  bit flips
- What if we chose  $n$  length codewords that are separated by sufficiently so that  $np$  bit flips does not hamper decodability?
- Number of such codewords is (Stirling's approximation):

$$\frac{2^n}{\binom{n}{np}} \simeq \frac{2^n}{\frac{n^n}{(np)^{np}(n-np)^{n-np}}} = 2^n p^p (1-p)^{1-p}$$

- Rate, in bits per symbol is:

$$\frac{1}{n} \log_2(2^n p^p (1-p)^{1-p}) = 1 + p \log_2 p + (1-p) \log_2 (1-p)$$

5

18:20 / 35:36

Let's illustrate this with  $n$ -bit error patterns. For example, if we consider zero errors, the total probability of observing zero errors is given by  $(1 - p)^n$ . For one error, the probability is  $n \cdot p \cdot (1 - p)^{n-1}$ . To be more formal, this can be expressed as:

$$\binom{n}{1} \cdot p^1 \cdot (1 - p)^{n-1}$$

Here, it is crucial to note that the term  $n \cdot p$  might not be negligible. In fact, due to the presence of  $(1 - p)^n$  and  $n \cdot p \cdot (1 - p)^{n-1}$ , a large  $n$  can make the single-error patterns more probable than the zero-error patterns.



Although the zero-error pattern is the most probable, its probability  $(1 - p)^n$  approaches zero as  $n$  approaches infinity, provided that  $p$  is non-zero. This means that with very large block lengths, it becomes nearly certain that we will not encounter an all-zero error pattern.

Thus, what we are seeking is the most probable group of error patterns for large block lengths. Let us denote the number of errors by  $l$  (as using  $k$  might be confusing due to its other uses). The probability of having exactly  $l$  errors is:

$$\binom{n}{l} \cdot p^l \cdot (1 - p)^{n-l}$$

This expression will help us identify the most likely error patterns in the given context.

Let's delve into the probability of encountering exactly  $l$  errors in a sequence. To determine this, we calculate the probability for each possible error sequence and sum them up. Given that these sequences are mutually exclusive, the total probability for having  $l$  errors is:

$$\binom{n}{l} \cdot p^l \cdot (1 - p)^{n-l}$$

This expression might exceed the probability of having zero errors,  $(1 - p)^n$ . Therefore, as the block length increases, it is essential to identify the most likely error sequences. This is the crux of our investigation.

To provide some intuition, consider a biased coin with the probability of heads being  $p$  and tails  $1 - p$ . The probability of obtaining a sequence of  $n$  heads is  $(1 - p)^n$ . If we seek the probability of getting exactly  $l$  tails (and thus  $n - l$  heads), it is given by:

$$\binom{n}{l} \cdot p^l \cdot (1 - p)^{n-l}$$

Now, if  $n$  is very large, say 10,000 or even 1 million, what value of  $l$  will maximize this probability? This situation aligns with the binomial distribution, where we seek the peak of the probability mass function (PMF). The peak, or the most likely number of tails (or heads), occurs when  $l$  is approximately  $np$ . This result is derived from the binomial distribution, where the maximum probability is found at  $l = np$ .

For a more intuitive understanding, consider the binomial distribution as the sum of independent Bernoulli trials. According to the Central Limit Theorem, the sum of a large number of independent Bernoulli random variables approaches a Gaussian distribution. Thus, if you plot the PMF of a binomial distribution for large  $n$ , it resembles a Gaussian shape with the peak near  $np$ .

In practical terms, knowing the most likely number of errors allows us to design codes effectively. For example, with Hamming codes, the goal is to correct errors by ensuring that code words are distinguishable even when a certain number of bits are flipped. Understanding the error distribution helps in designing codes that handle the expected number of errors, thus addressing the capacity aspect of coding theory.

The reason we use the term "approximate" is that  $k$  is an integer, and  $np$  may or may not be an integer. Nonetheless, the peak of the binomial distribution consistently occurs close to  $np$ . This indicates that the most likely number of errors is around  $np$ . In other words, the most probable error pattern involves approximately  $np$  bit flips. However, the challenge is that we do not know the exact locations of these  $np$  bit flips. If we did, it would simplify things considerably.

The goal is to design your code so that, even if  $np$  bits are flipped, the code can still map back to the original code word rather than being misinterpreted as a different one. The code should be structured such that flipping up to  $np$  bits will not result in a decoding error, ensuring the distance between code words remains sufficient.

Let's explore this concept further. Suppose we use  $n$ -length code words and arrange them such that  $np$  bit flips do not affect decodability. For instance, with Hamming codes, we designed the code words to be separated enough so that a single bit flip wouldn't cause decoding issues. Now, we are aiming to ensure that  $np$  bit flips are also manageable.

Here, we introduce a packing-based argument. The total number of possible  $n$ -bit sequences is  $2^n$ , where  $n$  is a large number. If we arrange  $n$ -bit code words such that flipping  $np$  bits still maintains unique decodability, we need to consider how many such sequences are possible.

To illustrate, consider a specific  $n$ -length code word. How many distinct  $n$ -length vectors can be generated by flipping  $np$  bits? It turns out that the number of such possible vectors is approximately  $\binom{n}{np}$ .

To put it simply, if you flip  $np$  bits in any  $n$ -length code word, the number of different resulting sequences is  $\binom{n}{np}$ . Therefore, the number of code words that can be accommodated within the  $2^n$  space is  $2^n$  divided by the volume of the "ball" with a size of  $\binom{n}{np}$ . This ball represents all possible sequences that result from flipping  $np$  bits.

Let me provide some intuition on this concept. Consider a space of  $2^n$   $n$ -bit sequences. Suppose you have a code word, which we'll call  $x$ . Any sequence obtained by flipping  $np$  bits in  $x$  will generate several other  $n$ -bit sequences. We need to ensure that these sequences don't overlap or get confused with other code words, such as  $x_1$ ,  $x_2$ , and so on. Each code word should have a distinct sphere around it, where the sphere's radius is defined by  $np$  bit flips. This way, no two code words should be indistinguishable from each other after  $np$  bit flips.

The challenge here is to pack these spheres into the  $2^n$  bit sequence space without overlap or confusion. To estimate how many such code words you can support, we calculate  $2^n$  divided by the number of possible sequences within each sphere, which is  $\binom{n}{np}$ .

Why  $\binom{n}{np}$ ? Because each sphere that accounts for  $np$  bit flips contains  $\binom{n}{np}$  possible sequences. Given that  $np$  bit flips are the most likely, we want to ensure that  $np$  bit flips do not lead to confusion. Thus, the approximate number of  $n$ -length code words you can support is:

$$\frac{2^n}{\binom{n}{np}}$$

Here, I've used Stirling's approximation to estimate  $\binom{n}{np}$ . Stirling's approximation simplifies  $n!$  for large  $n$  and is proportional to  $n^n$ . Using this approximation, we find:

$$\binom{n}{np} \approx \frac{n^n}{(np)^{np} \cdot (n - np)^{n - np}}$$

If  $np$  is not an integer, bear with me for a moment and assume it is close enough to an integer. Then:

$$\frac{2^n}{\frac{n^n}{(np)^{np} \cdot (n - np)^{n - np}}}$$

Simplifies to:

$$2^n \cdot \frac{(np)^{np} \cdot (n - np)^{n - np}}{n^n}$$

(Refer Slide Time: 24:09)

Week 12: Lecture 59

$$\frac{2^n}{\binom{n}{np}}$$

$$R = 1 + p \log_2 p + (1-p) \log_2 (1-p)$$

Cases:

$p = 0 \Rightarrow R = 1$

$p = \frac{1}{2} \Rightarrow R = 1 + \frac{1}{2} \log_2 \frac{1}{2} + \frac{1}{2} \log_2 \frac{1}{2}$

$= 1 + \log_2 \left(\frac{1}{2}\right)$

24:09 / 35:36

When you take the numerator and simplify,  $n^n$  cancels out, leaving:

$$2^n \cdot p^{np} \cdot (1 - p)^{n - np}$$

This result represents the number of code words. For a block length  $n$  and  $n$  uses of the

binary symmetric channel, the number of code words you can transmit is:

$$2^n \cdot p^p \cdot (1 - p)^{n-p}$$

The rate of information transmission is found by dividing the number of code words by  $n$  and taking the base-2 logarithm. The reason for using base-2 logarithm is straightforward: if you have 4 code words, you can encode 2 bits; with 8 code words, you can encode 3 bits; and with  $m$  code words, you can encode  $\log_2(m)$  bits.

Here's why we use logarithms and division by  $n$  to determine the rate per channel use. By taking the base-2 logarithm of our result and dividing by  $n$ , we obtain the rate for each use of the binary symmetric channel. This results in the formula:

$$1 - p \log_2(1 - p) - p \log_2(p)$$

This formula represents the rate achievable over a binary symmetric channel. To get a sense of this, let's calculate some examples.

When  $p = 0$ , we encounter an issue with  $p \log_2(p)$ . Since  $\log_2(0)$  tends to negative infinity, we define  $p \log_2(p)$  as 0 for this case. Thus, the rate  $r$  simplifies to 1 when  $p = 0$ , meaning that the channel can perfectly transmit one bit per use with no errors.

Now, let's consider  $p = 0.5$ , which represents a fair coin toss scenario. In this case, we compute:

$$r = 1 + (0.5 \log_2(0.5) + 0.5 \log_2(0.5))$$

Since  $\log_2(0.5) = -1$ , this simplifies to:

$$r = 1 + (0.5 \cdot (-1) + 0.5 \cdot (-1))$$

$$r = 1 - (-1)$$

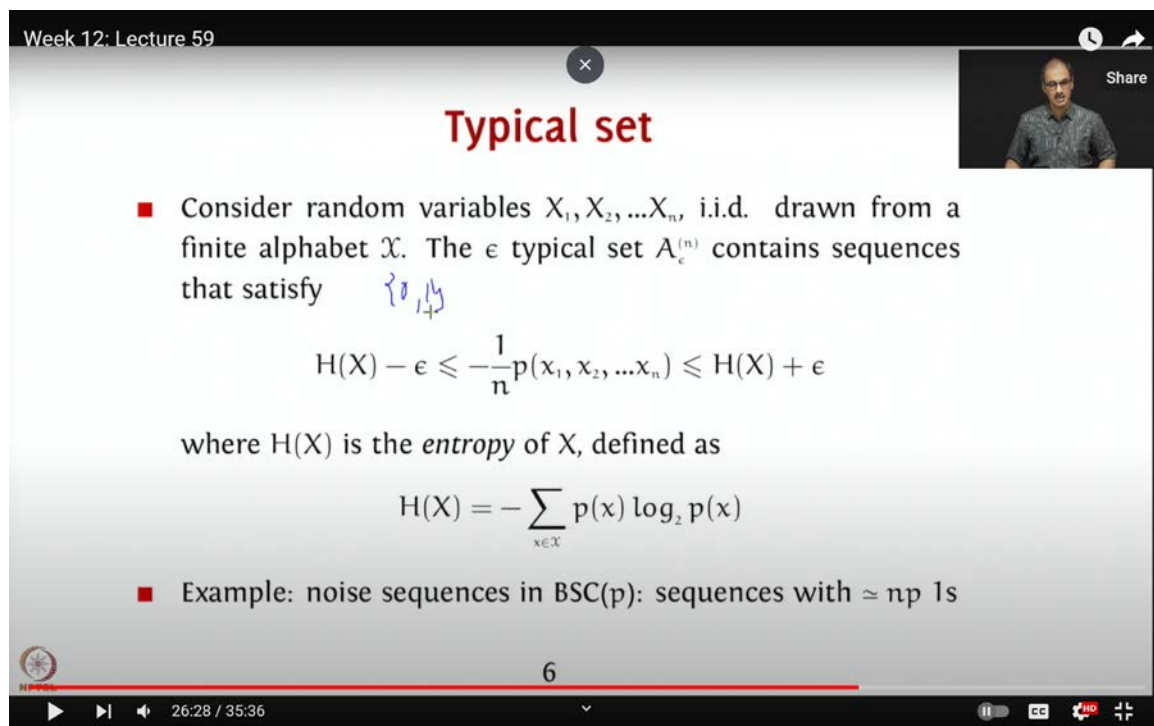
$$r = 1 - 1$$

$$r = 0$$

This means the achievable rate is 0 when  $p = 0.5$ . What does this imply? When  $p = 0.5$ , errors occur with equal probability for heads and tails. If you have an equal number of heads and tails (or  $n/2$  errors), distinguishing between different code words becomes nearly impossible. For example, if you use a repetition code with  $n$  bits and flip  $n/2$  bits, you end up with an equal number of ones and zeros, making it impossible to determine the original code word.

Thus, the rate being 0 when  $p = 0.5$  signifies that even a repetition code is not effective in this scenario because the channel is too noisy for reliable communication.

(Refer Slide Time: 26:28)



Week 12: Lecture 59

## Typical set

- Consider random variables  $X_1, X_2, \dots, X_n$ , i.i.d. drawn from a finite alphabet  $\mathcal{X}$ . The  $\epsilon$  typical set  $A_\epsilon^{(n)}$  contains sequences that satisfy  $\{0, 1\}$ 

$$H(X) - \epsilon \leq -\frac{1}{n} \log_2 p(x_1, x_2, \dots, x_n) \leq H(X) + \epsilon$$

where  $H(X)$  is the *entropy* of  $X$ , defined as

$$H(X) = - \sum_{x \in \mathcal{X}} p(x) \log_2 p(x)$$
- Example: noise sequences in BSC( $p$ ): sequences with  $\approx np$  1s

6

26:28 / 35:36

When  $p = 0$ , however, the channel is perfect, no bit flips occur, so the rate is 1 bit per channel use. This reflects a situation where the binary symmetric channel perfectly transmits zeros as zeros and ones as ones.

Let's delve into the concept of the "typical set," which formalizes the earlier statements we've made about probability. Consider a sequence of random variables  $x_1, x_2, \dots, x_n$ , which are independently and identically distributed (i.i.d.) and drawn from a finite alphabet

X. The typical set  $A_\epsilon^n$  is defined to include sequences that satisfy:

$$H(x) - \epsilon \leq -\frac{1}{n} \log_2 P(x_1, x_2, \dots, x_n) \leq H(x) + \epsilon,$$

where  $H(x)$  represents the entropy of  $x$ , defined as:

$$H(x) = - \sum_{x \in X} p(x) \log_2 p(x).$$

In the binary case where our alphabet consists of  $\{0,1\}$ ,  $H(x)$  corresponds to the binary entropy function. The expression  $H(x) - \epsilon \leq -\frac{1}{n} \log_2 P(x_1, x_2, \dots, x_n) \leq H(x) + \epsilon$ , indicates that the probability of observing a typical sequence of errors will lie between  $2^{-n(H(x)+\epsilon)}$  and  $2^{-n(H(x)-\epsilon)}$ .

(Refer Slide Time: 29:43)

Week 12: Lecture 59

## BSC Capacity

- As  $n \rightarrow \infty$ , typical set sequences amass all the probability
- Therefore, design code to account for this; partition  $2^n$  sequences into those that are robust to typical error sequences
- Formal definition: if  $X^n$  is sent and  $Y^n$  is the noisy output from a BSC(p), then the capacity is (using  $H(\cdot)$ ):

$$\simeq \frac{1}{n} \log_2 \left( \frac{2^{nH(Y)}}{2^{nH(Y|X)}} \right) = \frac{1}{n} \log_2 \left( \frac{2^{nH(Y)}}{2^{nH(N)}} \right) = 1 - H(p)$$

where  $H(p) = -p \log_2 p - (1-p) \log_2 (1-p)$

7

29:43 / 35:36

What this means in practice is that, for a sufficiently large  $n$  and for any given  $\epsilon$  (say  $\epsilon = 0.01$  or  $\epsilon = 0.001$ ), the probability of encountering the most likely error sequences is highly concentrated around the entropy  $H(x)$ . Essentially, typical sequences' probabilities will



always be less than or equal to  $2^{-nH(x)}$  and greater than  $2^{-n(H(x)+\epsilon)}$ .

To illustrate, if you take any sequence with  $n p$  ones in the context of a binary symmetric channel, this sequence will likely fall into the typical set. The principle here mirrors what we observed earlier: the number of ones  $l$  that maximizes the likelihood is  $n p$ . This concept is grounded in the weak law of large numbers, which tells us that, as  $n$  grows, sequences with  $n p$  ones become increasingly probable, and their probability approximates  $2^{-nH(x)}$ .

Thus, when designing codewords, you need to account for these typical error patterns, as they dominate the probability distribution. As  $n$  approaches infinity, the typical set sequences accumulate all the probability mass. Consequently, a sequence like the all-zero error sequence will have a probability approaching zero, reinforcing the idea that the typical error patterns are the ones you must be prepared to handle.

In terms of error sequences, sequences with zero errors have zero probability. However, sequences with errors close to  $n p$  have a non-zero probability. Each of these typical error sequences has a probability of approximately  $2^{-nH(x)}$ , and together, they account for almost all the probability mass. Therefore, you only need to focus on these typical error sequences.

To visualize this, consider that you have  $2^n$  possible sequences. Your goal is to partition these sequences so that they are robust against any errors of length  $n p$ . Essentially, you need to design your code so that the  $2^n$  sequences are organized in a way that they can handle typical error patterns.

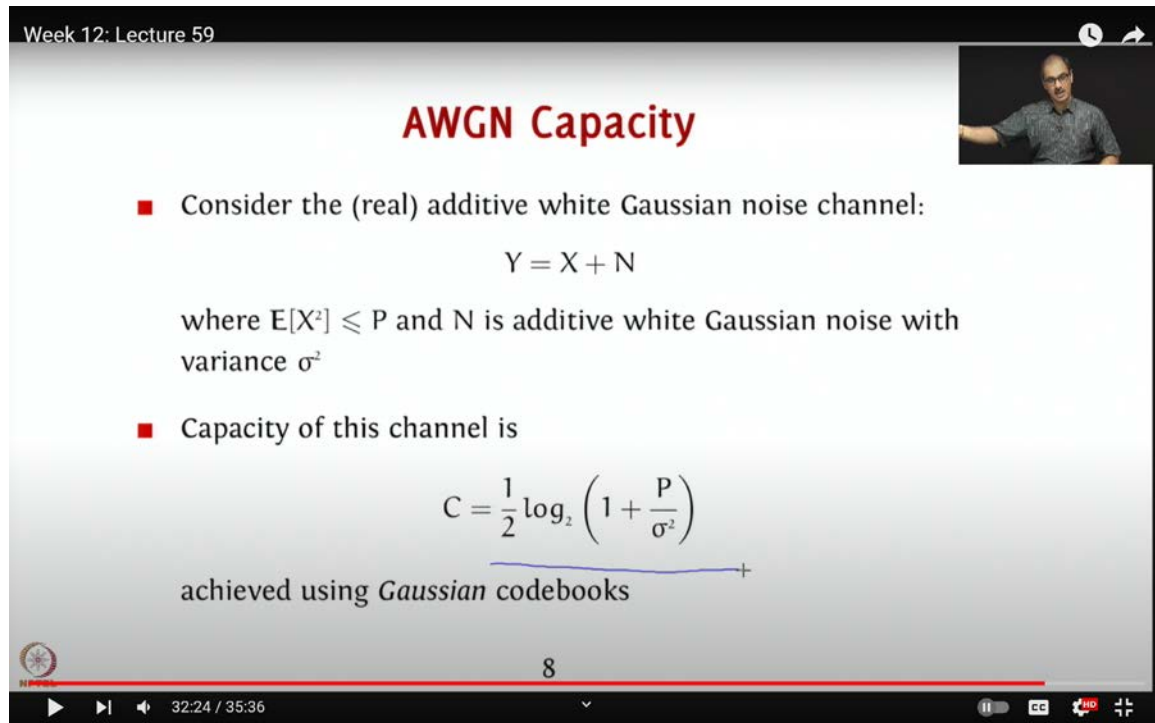
Formally, if you consider  $y = x + \text{noise}$ , where  $x$  is chosen from a uniform distribution of 0s and 1s and noise follows the binary symmetric channel distribution, you'll find that  $2^{nH(y)}$  relates to the capacity of the channel. Specifically, the capacity  $C$  of the binary symmetric channel is  $1 - H(p)$ , where  $H(p)$  is the binary entropy function, defined as:

$$H(p) = -p \log_2 p - (1 - p) \log_2(1 - p).$$

This matches the earlier expression  $1 + p \log_2 p + (1 - p) \log_2(1 - p)$ . This is the formal expression of the channel capacity, which can be validated through rigorous proofs in

textbooks such as "Fundamentals of Information Theory" by Cover and Thomas or other information theory resources.

(Refer Slide Time: 32:24)



The screenshot shows a video player interface for a lecture titled "Week 12: Lecture 59". The main content is a slide titled "AWGN Capacity" in red. The slide contains the following text:

- Consider the (real) additive white Gaussian noise channel:
$$Y = X + N$$
where  $E[X^2] \leq P$  and  $N$  is additive white Gaussian noise with variance  $\sigma^2$
- Capacity of this channel is
$$C = \frac{1}{2} \log_2 \left( 1 + \frac{P}{\sigma^2} \right)$$
achieved using Gaussian codebooks

A blue line underlines the formula for capacity, and a blue arrow points from the text "achieved using Gaussian codebooks" to the formula. The video player shows a progress bar at 32:24 / 35:36 and a small video inset of the lecturer in the top right corner.

The key insight is that for  $n$ -length bit sequences, you select  $2^k$  sequences so that any  $n$  bit flips do not lead to ambiguity. By focusing on typical error sequences and partitioning your codewords accordingly, you achieve this capacity.

As an aside, consider the additive white Gaussian noise (AWGN) channel. Here, the constraint is that the expectation of  $x^2$  must be less than or equal to  $P$ , where  $P$  is the power constraint. For an AWGN channel with Gaussian noise having zero mean and variance  $\sigma^2$ , the channel capacity is given by:

$$C = \frac{1}{2} \log_2 \left( 1 + \frac{P}{\sigma^2} \right).$$

This formula reflects that as  $P$  increases, the capacity increases, allowing you to use higher power and achieve higher rates. The number of possible  $x$  values you can choose relates to this capacity. Just as with the binary symmetric channel, you cannot directly use  $x$  values

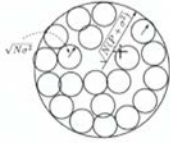
without accounting for noise.

(Refer Slide Time: 33:24)

Week 12: Lecture 59

## AWGN Capacity

- Intuition: use channel  $N$  times; “signal + noise” sphere has volume  $\simeq \left(\sqrt{N(P + \sigma^2)}\right)^N$ , noise sphere has volume  $\left(\sqrt{N\sigma^2}\right)^N$



- Therefore, achievable rate, based on a packing argument (can be shown to be equal to capacity), is

$$R = \frac{1}{N} \log_2 \left( \frac{\left(\sqrt{N(P + \sigma^2)}\right)^N}{\left(\sqrt{N\sigma^2}\right)^N} \right) = \frac{1}{2} \log_2 (1 + P/\sigma^2)$$

9

33:24 / 35:36

To achieve the channel capacity, you need to construct multiple codebooks and use the channel repeatedly. By doing so, you can leverage a large number of channel uses to reach the desired capacity through the concept of typicality.

Consider using the channel  $n$  times. Just like with the binary symmetric channel, you would have sequences  $x_1, x_2, \dots, x_n$ , and the noise would result in sequences like  $x_1 + n_1, x_2 + n_2$ , and so forth. Here,  $n$  represents the number of channel uses, and it directly affects how noise impacts the signal.

To visualize this, think of the signal-to-noise ratio as a sphere in  $n$ -dimensional space. The volume of this sphere is approximately proportional to  $\left(\sqrt{N} \cdot (p + \sigma^2)\right)^N$ , where  $p$  is the signal power and  $\sigma^2$  is the noise variance. The noise sphere has a volume proportional to  $\left(\sqrt{N} \cdot \sigma^2\right)^N$ . Ignoring the constants for simplicity, you can calculate the ratio  $r$  of these volumes as:

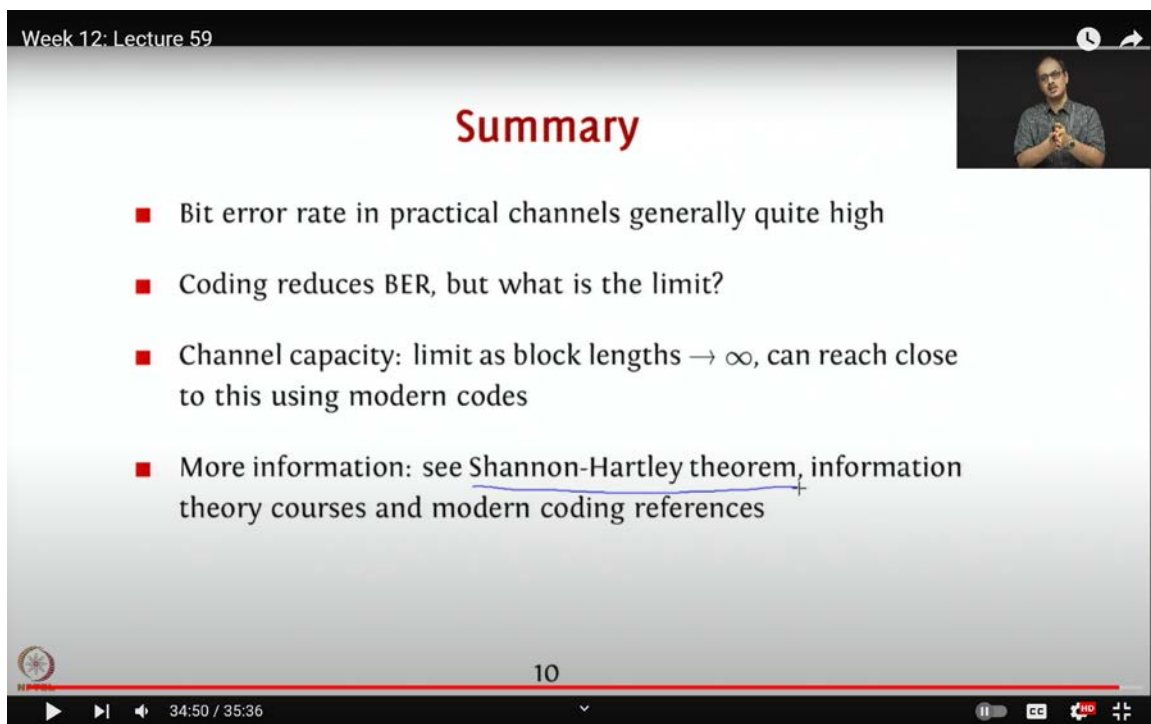
$$r = \frac{1}{N} \log \left( \frac{(\sqrt{N \cdot (p + \sigma^2)})^N}{(\sqrt{N \cdot \sigma^2})^N} \right).$$

Simplifying this, you get:

$$r = \frac{1}{2} \log_2 \left( 1 + \frac{p}{\sigma^2} \right).$$

This indicates that the channel capacity, or the maximum achievable rate, is  $\frac{1}{2} \log_2 \left( 1 + \frac{p}{\sigma^2} \right)$ . For a more detailed exploration, refer to a comprehensive information theory textbook.

(Refer Slide Time: 34:50)



Week 12: Lecture 59

## Summary

- Bit error rate in practical channels generally quite high
- Coding reduces BER, but what is the limit?
- Channel capacity: limit as block lengths  $\rightarrow \infty$ , can reach close to this using modern codes
- More information: see Shannon-Hartley theorem, information theory courses and modern coding references

10

34:50 / 35:36

In practical scenarios, the bit rate you can achieve is often lower than the theoretical capacity due to various factors that reduce efficiency. However, if you desire an arbitrarily low bit error rate, the channel capacity represents the ultimate limit. With sufficiently large block lengths, you can approach this limit. Modern coding techniques, such as LDPC (Low-Density Parity-Check) codes and turbo codes, are designed to get very close to this

capacity, although some error margin is still present.

For additional insights, consider reviewing resources that cover the Shannon-Hartley theorem, which extends these principles to channels with bandwidth. This theorem provides a way to calculate the achievable rate per Hertz in wideband channels. For more information on information theory and modern coding techniques, you can explore relevant textbooks and courses. Thank you.