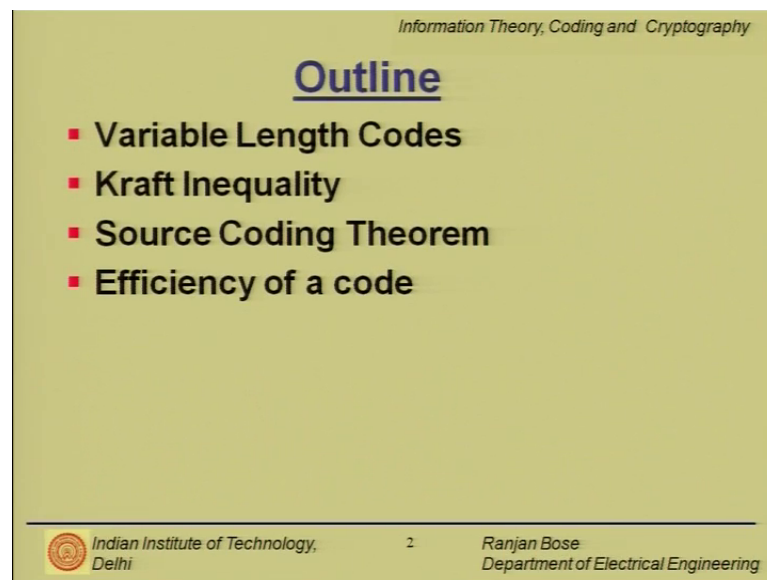


Information Theory, Coding and Cryptography
Dr. Ranjan Bose
Department of Electrical Engineering
Indian Institute of Technology, Delhi

Module - 04
Source Coding
Lecture - 04

Hello and welcome to module 4 on Source Coding. Let us look at the outline for today's presentation. We will start with variable length codes, then we will follow it up with Kraft inequality.


(Refer Slide Time: 00:36)



Information Theory, Coding and Cryptography

Outline

- Variable Length Codes
- Kraft Inequality
- Source Coding Theorem
- Efficiency of a code

 Indian Institute of Technology,
Delhi

2

Ranjan Bose
Department of Electrical Engineering


Then we will talk about the source coding theorem, and finally we will discuss what do we mean by the efficiency of a code?

(Refer Slide Time: 00:47)

Information Theory, Coding and Cryptography

Recap

- Information Measures for Continuous Random Variables
- Differential Entropy
- Average Conditional Entropy
- Relative Entropy (Kullback Leibler (KL) distance)
- Jensen Shannon distance
- Prefix Codes

 Indian Institute of Technology,
Delhi

3

Ranjan Bose
Department of Electrical Engineering

But let us start with a quick recap of what we have done and we will refresh our memories.

So, what we have done so far is we have looked at information measures for continuous random variables; wherein we looked at the notion of differential entropy, we will quickly revisit it again. Then we talked about average conditional entropy for continuous random variable, then we measured the similarity between 2 distributions in terms of relative entropy which is also called the Kullback Leibler distance. We looked at the asymmetry of that and then looked at Jensen Shannon distance. And finally, we introduced the notion of Prefix Codes.

(Refer Slide Time: 01:34)

Information Theory, Coding and Cryptography


Quick memory refresh

- The **self information** of the event $X = x_i$ is defined as

$$I(x_i) = \log\left(\frac{1}{P(x_i)}\right) = -\log P(x_i).$$

- The **average self information** of the event $X = x_i$ is defined as

$$H(X) = \sum_{i=1}^n P(x_i) I(x_i) = -\sum_{i=1}^n P(x_i) \log P(x_i)$$

 Indian Institute of Technology,
Delhi4Ranjan Bose
Department of Electrical Engineering

So, let us start a quick memory refresh. We started off with self information of the event x is equal to x_i , which we saw had a logarithmic measure for information. Then we said that it is important to have the average self information also called entropy defined as H of X right. So, it is with a negative sign $P(x_i) \log P(x_i)$ summed over all possible occurrences.

(Refer Slide Time: 02:08)

Information Theory, Coding and Cryptography


Quick memory refresh

- The **mutual information** between the event $X = x_i$ and $Y = y_j$ defined as

$$I(x_i; y_j) = \log\left(\frac{P(x_i | y_j)}{P(x_i)}\right) = \log\left(\frac{P(y_j | x_i)}{P(y_j)}\right) = I(y_j; x_i)$$

- The **average mutual information** between two random variables X and Y is given by

$$I(X; Y) = \sum_{i=1}^n \sum_{j=1}^m P(x_i, y_j) I(x_i; y_j) = \sum_{i=1}^n \sum_{j=1}^m P(x_i, y_j) \log \frac{P(x_i, y_j)}{P(x_i)P(y_j)}$$

 Indian Institute of Technology,
Delhi5Ranjan Bose
Department of Electrical Engineering

We also defined mutual information between x is equal to x_i and y is equal to y_j . So, between 2 events as x_i of x_i semicolon y_j and we found out that it is mathematically equivalent to the mutual information between y_j semicolon x_i .

This we realized could also be negative then we looked at the definition of average mutual information which is $I(X; Y)$ which is defined with a double summation and we also discovered that this quantity is non-negative. We then moved to continuous random variables where we defined the differential entropy denoted by small h of X as follows.

(Refer Slide Time: 02:55)

Information Theory, Coding and Cryptography

Quick memory refresh

- The **Differential Entropy** of a continuous random variable X is defined as

$$h(X) = - \int_{-\infty}^{\infty} p(x) \log p(x) dx$$

- Again, it should be understood that there is **no physical meaning** attached to the above quantity
- Chain Rule:

$$h(X_1, X_2, \dots, X_n) = \sum_{i=1}^n h(X_i | X_1, X_2, \dots, X_{i-1})$$

Indian Institute of Technology, Delhi
 6
Ranjan Bose
Department of Electrical Engineering

But we emphasized again and again that there is no physical meaning attached to it. So, I can plug in the value of p of x and compute it will come out to be finite, but we all know that the average self information for a continuous random variable is indeed infinite. Therefore, differential entropy really does not have a physical meaning attached to it. We also looked at the chain rule for differential entropy given as follows.

(Refer Slide Time: 03:44)

Information Theory, Coding and Cryptography


Quick memory refresh

- The **Relative Entropy** or **Kullback Leibler (KL) distance** between two probability mass functions $p(x)$ and $q(x)$ is defined as

$$D(p \parallel q) = \sum_{x \in X} p(x) \log \left(\frac{p(x)}{q(x)} \right)$$

- It can be interpreted as the expected value of $\log \left(\frac{p(x)}{q(x)} \right)$

Does not follow
the symmetry property or the triangle inequality.

 Indian Institute of Technology,
Delhi

7

Ranjan Bose
Department of Electrical Engineering

After that we asked ourselves a question how do you say how similar are 2 probability mass functions $p(x)$ and $q(x)$. And we defined the relative entropy also known as the Kullback Leibler distance $D(p \parallel q)$ as follows and it can be seen as simply the expected value of \log of $p(x)$ or $q(x)$.

So, it goes without saying that if $p(x)$ equals to $q(x)$ $\log 1$ is 0 and there is the distance reduces to 0; that is there is no difference between the distributions p and q . On the other hand quick observation of this equation tells us that this is not symmetric that is to say the $D(p \parallel q)$ is not the same as the $D(q \parallel p)$ this is important to note and therefore, it does not follow the symmetry property.

We also found out that this relative entropy or the Kullback Leibler distance, does not satisfy the triangle inequality. So, it does not follow 2 of the 3 requirements for a distance measure; it is only non negative. It does not follow the symmetry property or the triangle inequality.

(Refer Slide Time: 05:16)

Information Theory, Coding and Cryptography


Quick memory refresh

- The **Jensen Shannon distance** between two probability mass functions $p(x)$ and $q(x)$ is defined as

$$JSD(p \parallel q) = \frac{1}{2}D(p \parallel m) + \frac{1}{2}D(q \parallel m)$$

where $m = \frac{1}{2}(p + q)$

- If the base of the logarithm is 2, then, $0 \leq JSD(p \parallel q) \leq 1$
- Jensen Shannon distance is sometimes referred to as **Jensen Shannon divergence** or **Information Radius** in literature.

 Indian Institute of Technology, Delhi8Ranjan Bose
Department of Electrical Engineering


To overcome this problem of symmetry Jensen Shannon Distance was defined again it is the distance between 2 probability mass functions $p(x)$ and $q(x)$ and it is defined as follows. And we found out that if the base of the log is 2 then the Jensen Shannon distance lies between 0 and 1. In literature this Jensen Shannon distance is also called Jensen Shannon divergence or sometimes information radius, but why are we doing all of this?

(Refer Slide Time: 05:52)

Information Theory, Coding and Cryptography

Quick memory refresh

- A **code** is a set of vectors called **codewords**
- The **fixed length code** for the English alphabet suggests that each of the letters in the alphabet is equally important (probable) and hence each one requires 5 bits for representation.
- However, we know that some of the letters are less common (x, q, z etc.) while some others are more frequently used (s, t, e etc.).
- It appears that allotting equal number of bits to both the frequently used letters as well as not so commonly used letters is *not* an efficient way of representation (coding).

 Indian Institute of Technology, Delhi9Ranjan Bose
Department of Electrical Engineering

Well, our ultimate aim is to efficiently represent symbols, we would like to code them so, that we use minimum number of bits to represent the symbols. For that we defined codes and codewords; so, a code is a set of vectors called codewords. And we could have a fixed length code for example, for the English alphabet and since we have 26 letters about 5 bits should suffice because 2 raised to power 5 is 32.

So, we will have some additional codewords available, but a fixed length code should suffice for an English alphabet. However, there is a simple observation that we can make not all letters are equiprobable what does it mean? We should not really allocate equal number of bits for less probable as opposed to more frequently appearing letters.

So, there could be a better strategy than to allocate equal number of bits for all the letters in the English alphabet.

(Refer Slide Time: 07:09)

Information Theory, Coding and Cryptography

Example

- Suppose we have only the **first eight letters** of the English alphabet (A – H) in our vocabulary.
- The **fixed length code** for this set of letters would be

Letter	Codeword	Letter	Codeword
A	000	E	100
B	001	F	101
C	010	G	110
D	011	H	111

- A **variable length code** for the same set of letters can be

Letter	Codeword	Letter	Codeword
A	00	E	101
B	010	F	110
C	011	G	1110
D	100	H	1111

Indian Institute of Technology,
Delhi

10

Ranjan Bose
Department of Electrical Engineering

So, we start off with an example; we did it in the previous class as well we have only the first 8 letters in the English alphabet for this example A, B, C, D, E, F, G and H and the simplest in the most obvious thing is 8 letters no problem, 3 bits per letter we have 8 possible codewords. So, codewords are from 0 0 0 up to 1 1 1 and this set of codewords is called a code. So, in this table we have the fixed length code, but nobody stops us from making a variable length code and how do we do that? Well a particular example is as follows I have A being represented as 0 0 just 2 bits, B with 3 bits so and so, forth and since we ran out of 2 bit or 3 bit we could also say that G and H also have 4 bits.

So, it is clearly a variable length code it goes without saying that there are several possible variable length codes available for these 8 letters.

(Refer Slide Time: 08:26)


Information Theory, Coding and Cryptography

Example

- Suppose we have to code the series of letters:
"A BAD CAB".
- The fixed length and the variable length representation of the pseudo sentence would be

Fixed Length Code	000 001 000 011 010 000 001	Total bits = 21
Variable Length Code	00 010 00 100 011 00 010	Total bits = 18

- Note that the variable length code uses **fewer numbers of bits**
- This is because the letters appearing more frequently in the pseudo sentence are represented with fewer numbers of bits.

 *Indian Institute of Technology,
Delhi* 11 *Ranjan Bose
Department of Electrical Engineering*

So, let us take a toy example A BAD CAB we would like to encode it. So, if we use the fixed length code we can use it to encode it and we get 21 bits whereas, for the variable length code; we have only eighteen bits to represent A BAD CAB.

So, we have fewer number of bits actually representing the variable length code right. So, we would like to now see is this the only possibility or what else can be done for representing A BAD CAB ok; I am a request somebody to please close the door, ok. So, we get back to this example of A BAD CAB.

(Refer Slide Time: 09:25)

Information Theory, Coding and Cryptography


Example

- We look at yet another variable length code for the first 8 letters of the English alphabet

Letter	Codeword	Letter	Codeword
A	0	E	10
B	1	F	11
C	00	G	000
D	01	H	111

- This second variable length code *appears* to be more efficient in terms of representation of the letters.

Variable Length Code 1	00 010 00 100 011 00 010	Total bits = 18
Variable Length Code 2	0 1001 0001	Total bits = 9

 Indian Institute of Technology, Delhi 12 Ranjan Bose
Department of Electrical Engineering

So, let us try another possible variable length code. So, here we are trying to be more aggressive again we have A B C D up to H and instead of 2 bits or 3 bits. We start up with 0 1 0 0 0 1 and this appears to be even more efficient because we are using fewer number of bits to represent it.

So, just compare the variable length code 1 and variable length code 2 and we come up with this remarkable observation that the second variable length code actually needs half the number of bits for this particular case A BAD CAB. But let us see is this true is this good enough, is there a problem with this representation.

(Refer Slide Time: 10:19)


Information Theory, Coding and Cryptography

Uniquely decodable !

Letter	Codeword	Letter	Codeword
A	0	E	10
B	1	F	11
C	00	G	000
D	01	H	111

Variable Length Code 2 0 1001 0001 **Total bits = 9**

- However there is a problem with VLC2.
- Consider the sequence of bits 0 1001 0001 which is used to represent **A BAD CAB**
- We could regroup the bits in a different manner to have [0] [10][0][1] [0][0][01] which translates to **A EAB AAD**
- or we can decode the vector as [0] [1][0][0][1] [0][0][0][1] which stands for **A BAAB AAAB !**
- **Not uniquely decodable**

 Indian Institute of Technology, Delhi13Ranjan Bose
Department of Electrical Engineering

So, we come to the question of uniquely decodable codes even though the variable length code 2 had only 9 bits to represent A BAD CAB, when you come to the other side of the table and decode it you are in trouble. Because, simply we have several possible decoding you could have A EAB AAD or another decoding possible AB AAB because we really do not know when does one codeword end and one codeword begin.

So, in today's lecture we would like to see is there an efficient way to do variable length codes which are uniquely decodable.


(Refer Slide Time: 11:09)

Information Theory, Coding and Cryptography

Revisit VLC1 and VLC2

VLC1				VLC2			
Letter	Codeword	Letter	Codeword	Letter	Codeword	Letter	Codeword
A	00	E	101	A	0	E	10
B	010	F	110	B	1	F	11
C	011	G	1110	C	00	G	000
D	100	H	1111	D	01	H	111

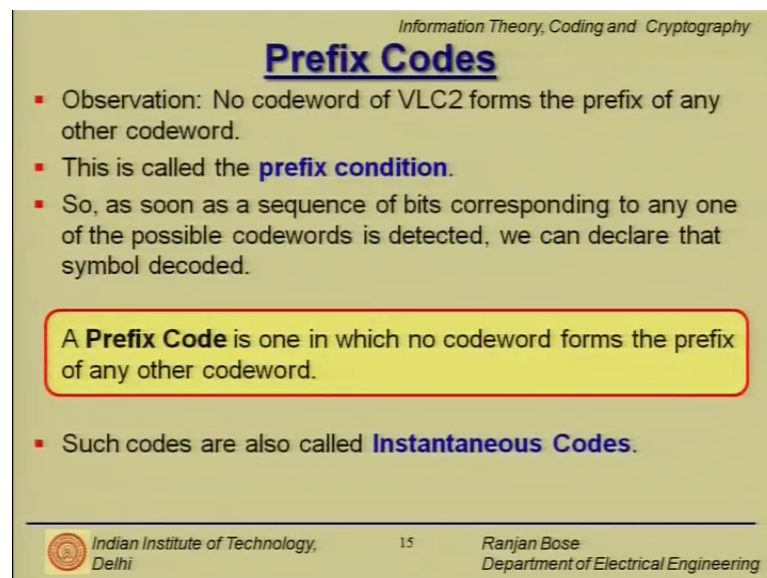
- **VLC2:** We have no clue where the codeword of one letter (symbol) ends and where the next one begins, since the lengths of the codewords are variable.
- However, this problem does not exist with the **VLC1**.
- **It can be seen that no codeword forms the prefix of any other codeword.**

 Indian Institute of Technology, Delhi14Ranjan Bose
Department of Electrical Engineering

So, let us revisit variable length code 1 and code 2 we established that with variable length code 2 unique decodability is not a possibility, but for variable length code 1; you can see that no codeword forms the prefix of any other codeword. So, let us emphasize that no codeword is a starting point of any other codeword. For example, if you consider A it is 0 0 you can go through the entire list; no codeword begins with 0 0. If you look at any other 0 1 0 for B no codeword starts with that.

Now, this means that no codeword is a prefix of any other codeword; consequently the moment we receive a codeword which is valid, which belongs to this set we declare it as received and we proceed further so decoding is possible.

(Refer Slide Time: 12:14)



The slide is titled "Prefix Codes" and is part of a presentation on "Information Theory, Coding and Cryptography". It contains the following text:

- Observation: No codeword of VLC2 forms the prefix of any other codeword.
- This is called the **prefix condition**.
- So, as soon as a sequence of bits corresponding to any one of the possible codewords is detected, we can declare that symbol decoded.

A **Prefix Code** is one in which no codeword forms the prefix of any other codeword.

- Such codes are also called **Instantaneous Codes**.

At the bottom of the slide, there is a footer with the Indian Institute of Technology Delhi logo, the page number 15, and the name of the lecturer, Ranjan Bose, from the Department of Electrical Engineering.

This condition is called prefix conditions and such codes where no codeword forms the prefix of any other codeword; they are called prefix codes. We also we saw that such codes can immediately give you the answer, give you the decoded symbol stream and hence they are called instantaneous codes.

(Refer Slide Time: 12:43)

Information Theory, Coding and Cryptography


Kraft Inequality

- A necessary and sufficient condition for the existence of a binary code with codewords having lengths $n_1 \leq n_2 \leq \dots \leq n_L$ that satisfy the **prefix condition** is

$$\sum_{k=1}^L 2^{-n_k} \leq 1$$

- For prefix codes over an alphabet of size M we have

$$\sum_{k=1}^L M^{-n_k} \leq 1$$

 Indian Institute of Technology, Delhi 16 Ranjan Bose
Department of Electrical Engineering


Now, let us look at a very interesting inequality called the Kraft inequality a necessary and sufficient condition for the existence of a binary code with codewords having lengths n_1 less than or equal to n_2 so and so, forth up to n_L which satisfies the prefix condition is for as follows summation over all possible codewords 2 raised power n minus k is less than or equal to 1 . This inequality is called Kraft inequality. For non binary case, for alphabet size of M . We have the same equivalent Kraft inequality with 2 being replaced by M as follows. We will use this extensively in our lecture.

(Refer Slide Time: 13:40)

Information Theory, Coding and Cryptography

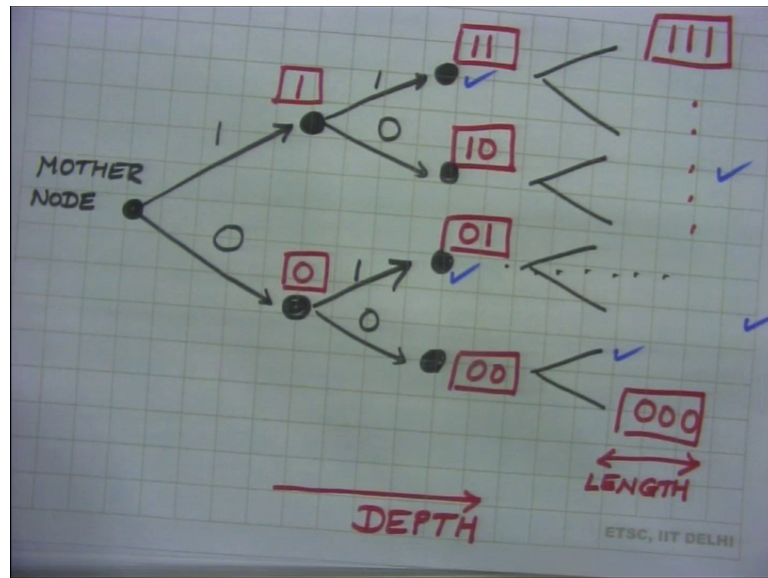
Example

- Consider the construction of a prefix code using a binary tree
- We start from the **mother node** and proceed toward the terminal nodes of the binary tree
- Let the mother node be labeled '0' (could have been labeled '1' as well).
- There are always two branches emanating from each node (binary tree).

 Indian Institute of Technology, Delhi 17 Ranjan Bose
Department of Electrical Engineering

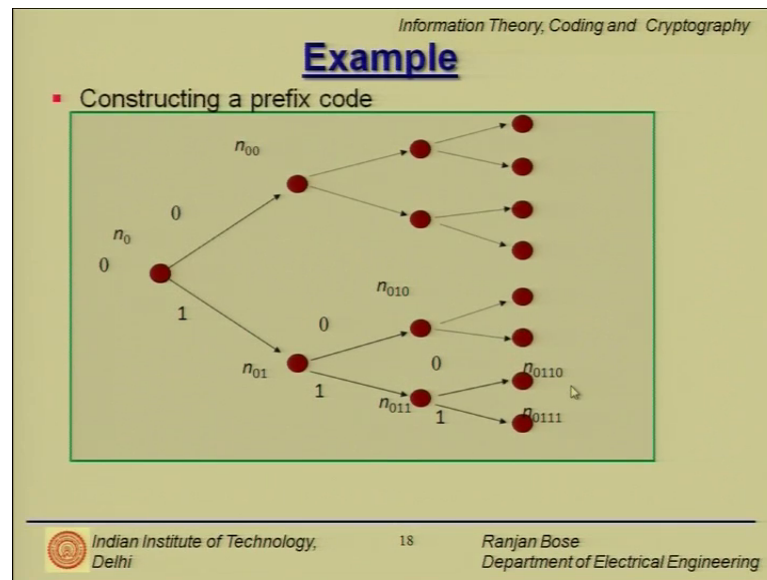
Now, let us consider the construction of a prefix code because they are great that instantaneous they uniquely decodable. So, it is better to find out how we can make or construct a prefix code using a binary tree. So, we start with a mother node and proceed towards terminal nodes of a binary tree. So, what is a binary tree? So, let us have a look at it we can go horizontal.

(Refer Slide Time: 14:12)



So, we can have a mother node and we can always have 2 daughters and either we take a 1 going up 0 going down and so and so forth. So, this is a binary tree and it can go on forever right.

(Refer Slide Time: 15:00)



So, there are always 2 branches emanating from a binary tree but let us take this a little further. And if we say this guy is 1 and this is 0 and then we go up here and we add 1 to this one. So, we get 1 1 and we can add 0 because we went down, we can add 1 0 and then 0 1 and 0 0. So, what are these? These are our options options for what; options for forming codewords because I can then go further down and have right from 1 1 1 going down up to 0 0 0. As we go down the depth of the tree, the length of the codeword increases; so, this is the possible length.


So, we are dealing with a variable length code. So, our problem of finding codeword tantamounts to choosing some of the nodes within this tree because each node is associated with a bit pattern which forms our codeword. So, let us understand the prefix code from this binary tree. So, let us look at this daughter node and from there we have; so, from the mother node we have 2 daughter nodes and then they multiply they grew exponentially and we have this neat looking binary tree.

(Refer Slide Time: 17:18)

Information Theory, Coding and Cryptography

Example

- Let us label the upper branch '0' and the lower branch '1' (these labels could have also been mutually exchanged).
- First we follow the upper branch from the mother node. We obtain our first codeword $c_1 = 0$ terminating at node n_{00} .
- Since we want to construct a prefix code where no codeword is a prefix of any other codeword, we must discard all the daughter nodes generated as a result of the node labeled c_1 .

 Indian Institute of Technology, Delhi 19 Ranjan Bose
Department of Electrical Engineering

Now, comes the question of labeling of the branches; so, it is our choice we can label the upper branch 0 lower branch 1 or we can exchange it without loss of generality and we can label the upper branch 1 and the lower branch 0 and then we can label the nodes as being codewords right. So, we have labeled in the previous one n_{00} , n_{01} , n_{10} , n_{11} , n_{110} .


So, basically n_{00} happens to be the mother node and whatever branch we are taking; we add that branch label to the name of the node. So, this is n_{01} and so on and so forth I went up over a 0 branch I added n_{010} and so and so forth.

(Refer Slide Time: 18:14)

Information Theory, Coding and Cryptography

Example

- We now proceed on the lower branch from the mother node and reach the node n_{01} .
- We proceed along the upper branch first and reach node n_{010} .
- We label this as the codeword $c_2 = 10$ (the labels of the branches that lead up to this node travelling from the mother node).
- Following the lower branch from the node n_{01} , we ultimately reach the terminal nodes n_{0110} and n_{0111} , which correspond to the codewords $c_3 = 110$ and $c_4 = 111$ respectively.
- Thus the binary tree has given us four prefix codewords: $\{0, 10, 110, 111\}$.
- By construction, this is a prefix code.

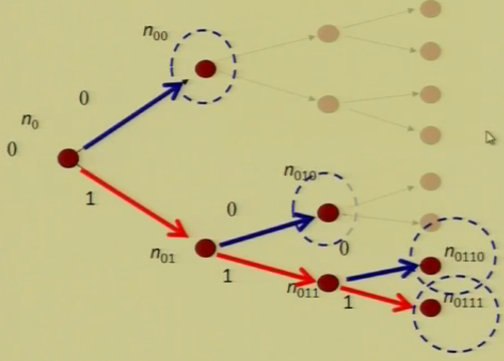
 Indian Institute of Technology, Delhi 20 Ranjan Bose
Department of Electrical Engineering


So, we keep going and we follow it and we can now have a possibility of understanding how to have a prefix code?

(Refer Slide Time: 18:21)

Information Theory, Coding and Cryptography

Example



 Indian Institute of Technology, Delhi 21 Ranjan Bose
Department of Electrical Engineering

So, what is the constraint for a prefix code? No codeword is a prefix of any other codeword. So, let us start somewhere; suppose we have 4 symbols to encode; so, we go to the first daughter node of course, we could have also gone on this side and let us declare this one n_{00} has one of the valid codewords.

Now, what we observe in a binary tree is that whenever we go below on to the further depths of the tree; we keep adding is 1 0 0 1, but since we are adding we lose the condition of being prefix. So, if you have declared the node n 0 0 as a valid codeword no daughter nodes below it can be valid nodes for codewords. So, we have to kind of wipe them off the screen and make sure that we never use any of these daughter nodes here because we have declared this one.

So, how the tree is gone? But we have the other half left since we would like to search the prefix codes within these trees. So, we proceed further because of the 4 codewords we want we have only found one. So, we proceed to the next level and we declare another valid codeword as belonging to the node n 0 1 0. Once we declare it the brief prefix condition requires us to declare any of the daughter nodes from this point onwards invalid.

So, we again kind of wipe them out from the board and once we have declared these 2 as valid codewords; we have nothing left here. But then we still have the quarter of the tree left and we proceed further and with a sign of relief we see that we still have 2 nodes left and so, we declare these 4 as valid codewords.

Please note if we took encode 5 symbols we could not declare the results right at this stage we would have to go deeper into the tree. The point to be noted in this example is that all valid prefix codes can be picked up from this tree. And I can easily extend this example to an M-ary tree for non binary codes, but let us just focus on binary codes here.

(Refer Slide Time: 21:19)

Information Theory, Coding and Cryptography

Example

$$\sum_{k=1}^L 2^{-n_k} = 2^{-1} + 2^{-2} + 2^{-3} + 2^{-3} = 0.5 + 0.25 + 0.125 + 0.125 = 1.$$

Hence the Kraft inequality is satisfied

Indian Institute of Technology, Delhi
22
Ranjan Bose
Department of Electrical Engineering

Now, the question is about the Kraft inequality does it really hold and we can check it. So, the Kraft inequality if you go back and see it is an upper bound summation over all possible codewords; 2 raised power minus n_k what is n_k ? It is the length of the code the length of the codeword because it is a variable length code. So, different codewords will have different lengths and the summation should be bounded by 1 .

So, we let us check this case for the code we just constructed. So, within this binary tree we have constructed a prefix code having 4 codewords and they are 0 10 110 and 111 . So length equal to 1 length equal to 2 3 and 3 ; so, if you substitute it in this Kraft inequality: so, we substitute the length of n_k . So, 1 2 for the second 3 and 3 we add it up you get 1 ; so, the inequality is satisfied.


(Refer Slide Time: 22:36)

Information Theory, Coding and Cryptography

Source Coding Theorem

- Let X be the ensemble of letters from a DMS with finite entropy $H(X)$ and the output symbols x_k , $k = 1, 2, \dots, L$, occurring with probabilities $P(x_k)$, $k = 1, 2, \dots, L$.
- It is possible to construct a code that satisfies the prefix condition and has an **average length** that satisfies the inequality

$$H(X) \leq \bar{R} < H(X) + 1.$$

 Indian Institute of Technology,
Delhi23Ranjan Bose
Department of Electrical Engineering

Now, we come to a very important theorem called the source coding theorem ok. It deals directly with the amount of compression we can achieve; so, let x be the ensemble of letters from a discrete memoryless source with finite entropy H of X . And the output symbols be x_k where k is equal to 1 2 3 up to L ; so, we have got L distinct symbols and these symbols are not equiprobable. Please note we have already guessed intuitively, if the symbols are equiprobable compression is not likely why? Because we have to allocate equal number of bits to all of the symbols, but only in the cases when the symbols are non equiprobable; we have the possibility to tweak.

So, the source coding theorem tells us that it is possible to construct a code that satisfies the prefix condition and also has an average length that satisfies the following inequality. This \bar{R} is the average length and it says that the average length for this for the symbols from this DMS is lower bounded by the entropy of the source H of X and is upper bounded by H of X plus 1; please note units are bits. So, it has far reaching consequences; it tells you the absolute limit to which I can compress the symbols emanating from the source.

(Refer Slide Time: 24:26)

Information Theory, Coding and Cryptography

Source Coding Theorem


- First consider the lower bound of the inequality.

$$H(X) - \bar{R} = \sum_{i=1}^L P(x_i) \log_2 \frac{1}{P(x_i)} - \sum_{i=1}^L P(x_i) n_i = \sum_{i=1}^L P(x_i) \log_2 \frac{2^{-n_i}}{P(x_i)}$$

We now make use of the inequality $\ln x \leq x - 1$ to get

$$\begin{aligned} H(X) - \bar{R} &\leq (\log_2 e) \sum_{i=1}^L P(x_i) \left(\frac{2^{-n_i}}{P(x_i)} - 1 \right) \\ &\leq (\log_2 e) \left(\sum_{i=1}^L 2^{-n_i} - 1 \right) \leq 0. \end{aligned}$$

The last inequality follows from the Kraft inequality. Equality holds if and only if $P(x_k) = 2^{-n_k}$ for $1 \leq k \leq L$. Thus the lower bound is proved.

 Indian Institute of Technology, Delhi
24
Ranjan Bose
Department of Electrical Engineering

So, let us quickly look at this theorem and consider the lower bound of the inequality. So, in order to prove that let us start with H of X minus R bar. So, we are trying to prove the left hand side H of X less than or equal to R bar to do so, we start with H of X minus R bar and we substitute this values. So, H of X is nothing but $\sum P(x_k) \log \frac{1}{P(x_k)}$ and R bar which is the average codeword length is nothing but $\sum P(x_k) n_k$; the length of the codeword x_k and multiplied with the probability $P(x_k)$ and this gives the average codeword length; you group them together and you get this expression.

Now, we have another inequality $\ln x \leq x - 1$. So, we would like to substitute it, but before that we must change the base of the log. So, we get outside log to the base to e and this log gets replaced by ln. So, this ln and this quantity is x. So, we replace $\ln x$ by $x - 1$ with an inequality. So, here you get the inequality and you get $x - 1$ alright. And then if you take this P of x and multiply it out and carry out the summation clearly this minus 1 with P of x all probabilities added up adds up to 1 and $\sum P(x_k) 2^{-n_k}$; $\sum P(x_k)$ cancels out and you have this summation $\sum 2^{-n_k} - 1$ is equal to $\sum 2^{-n_k} - 1$ and this is clearly less than 0 from the Kraft inequality.

So, what is less than or equal to 0? $H(X) - \bar{R}$ is less than or equal to 0 and consequently you have this left hand side $H(X) \leq \bar{R}$. Now let us consider the upper bound; so, first let us select codewords of length n_k such that $2^{-n_k} \leq P(x_k) \leq 2^{-n_k + 1}$.

raised power $n_k + 1$. So, we are hoping that we can do so, in due course we will find out that it indeed we can do so.

So, just consider the left hand side of this what are we considering? 2^{n_k} less than or equal to $P(x_k)$. Now summing both sides from k is equal to L through 1 right we get that indeed this is the Kraft inequality and hence whatever we thought in the beginning we can do. So now, consider the right hand side $P(x_k)$ less than or equal to $2^{n_k + 1}$ we take log on both sides. So, it becomes log to the base 2 $P(x_k)$ and of course, log to the base 2. Hence, this 2 goes away and we have $n_k + 1$ and we reshuffle it and we can write n_k is less than equal to $1 + \log_2 P(x_k)$.

Now, what do we do with this quantity? Well we multiply this both sides with $P(x_k)$. So, we have $P(x_k)$ multiplied with n_k here $P(x_k)$ minus $P(x_k) \log_2 P(x_k)$.

(Refer Slide Time: 28:07)

Information Theory, Coding and Cryptography

Source Coding Theorem

- On multiplying both sides by $P(x_k)$ and summing over $1 \leq k \leq L$ we obtain


$$\sum_{k=1}^L P(x_k) n_k < \sum_{k=1}^L P(x_k) + \left(- \sum_{k=1}^L P(x_k) \log_2 P(x_k) \right),$$

or,

$$\bar{R} < H(X) + 1.$$

- Thus

$$H(X) \leq \bar{R} < H(X) + 1.$$

 Indian Institute of Technology,
Delhi

26

Ranjan Bose
Department of Electrical Engineering

And if we sum it over all k is equal to 1 through L ; on the left hand side we have $\sum_{k=1}^L P(x_k) n_k$ summation. Again summation $\sum_{k=1}^L P(x_k)$ and we have this summation. But suddenly we can see a very simple formula emerging from it this left hand side is nothing but the average codeword length \bar{R} and it is less than this quantity which is 1 plus this is the entropy.

So, you can show the upper bound for the source coding theorem. So, lower bound was proved using the Kraft inequality and we also use the Kraft inequality in parts for the

upper bound. So, what is the implication of the source coding theorem? First and foremost remember we use the condition of the prefix code.

(Refer Slide Time: 29:00)

Information Theory, Coding and Cryptography

Source Coding Theorem

- The theorem tells us that for any **prefix code** used to represent the symbols from a source:
 - the *minimum* number of bits required to represent the source symbols
 - on an average
 - must be at least equal to the entropy of the source.

Indian Institute of Technology, Delhi 27 Ranjan Bose Department of Electrical Engineering

So, the source coding theorem deals with codes that satisfy the prefix condition ok. So, what does it tell us? It tells us something about the minimum number of bits required to represent source symbol, it gives us the lower bound and upper bound.

So, it says that the minimum number of bits required to represent source symbols on an average because remember we have used H of X which is the average self information. So, always it will never happen, but on an average the minimum number of bits required to represent the source symbol must be at least equal to the entropy of the source. That means, a source with a larger entropy would require larger of number of bits to represent it; it fits in very well intuitively.

But we also have this upper limit the upper limit says that if you are using prefix code then you are not too far away from the best you can do. What is the best you can do? H of X you are only 1-bit off; so, even though a prefix code may or may not be able to achieve the lower bound, it will never be too far away from the best you can do; prefix codes are good that is the take home message wherever possible use prefix code.



(Refer Slide Time: 30:54)

Information Theory, Coding and Cryptography


Source Coding Theorem

- If we have found a prefix code that satisfies $\bar{R} = H(X)$ for a certain source X , we must abandon further search because we cannot do any better.
- The theorem also tells us that a source with higher entropy (uncertainty) requires, on an average, more number of bits to represent the source symbols in terms of a prefix code.

Source A



Source B

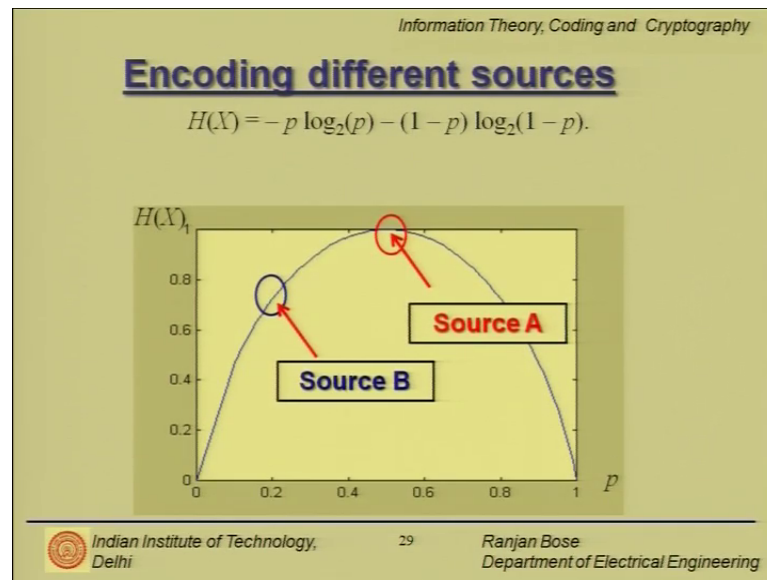
 Indian Institute of Technology, Delhi28Ranjan Bose
Department of Electrical Engineering

So, it also tells us that if we have found a prefix code that already satisfies this condition \bar{R} is equal to $H(X)$; abandon further search you cannot better this, you have hit rock bottom, you cannot compress it any further. And this also tells us that a source with higher entropy, uncertainty requires on an average more number of bits to represent the source symbols.

So, let us take an example; I have source A and I have source B; what is my source A? We take our favorite example a person tossing a coin and let us say he is tossing a fair coin. So, 1 0 0 1 how does he give these answers? Head speaks out one tail speaks out 0 and keeps tossing that is his only aim in life tossing a coin and calling out 1 1 0 1 0 and so forth, but there is this guy B called source B habit of chewing on a gum; sticks the gum on one side the coin becomes weighted unfair coin still tosses the coin and he reads out 1 1 0 0 0 1 and so and so forth.

But these 2 sources even though are emanating 1-bit say every 1 second should not required equal number of bits to present the outcomes because source coding theorem says that look the entropies are different.

(Refer Slide Time: 32:47)



So, we go back to our entropy map; so, binary entropy function on the x axis we have p the probability of occurrence of head and on the y axis we have the average self information entropy H of X and we less mark the position of source A.

So, source A who is tossing a fair coin sits at the top of the curve with H of X is equal to 1 bit, but my chewing gum friend source B who is tossing a biased coin sits somewhere low. So, source coding theorem tells us that if we figure out a way to efficiently represent the source B symbols, we only need roughly 0.7 bits on an average not 1, but here is the dilemma source B every second tosses the coin and shouts 1 0 0 0 1, but he should actually be shouting information equal into 0.7 bits. He is using the entire 1-bit; he has to do things differently. Maybe he has to toss 2 times and then speak out the output in an encoded manner or possibly toss 3 times and whatever is the observed output should speak out in terms of an encoded bit stream.


So, clearly source coding theorem tells us that hey it is unfair to allocate 1-bit per second to source A and also 1-bit per second to source B where these are the 2 guys tossing a fair and unfair coin once every second.

(Refer Slide Time: 35:00)

Information Theory, Coding and Cryptography

Efficiency of a Code

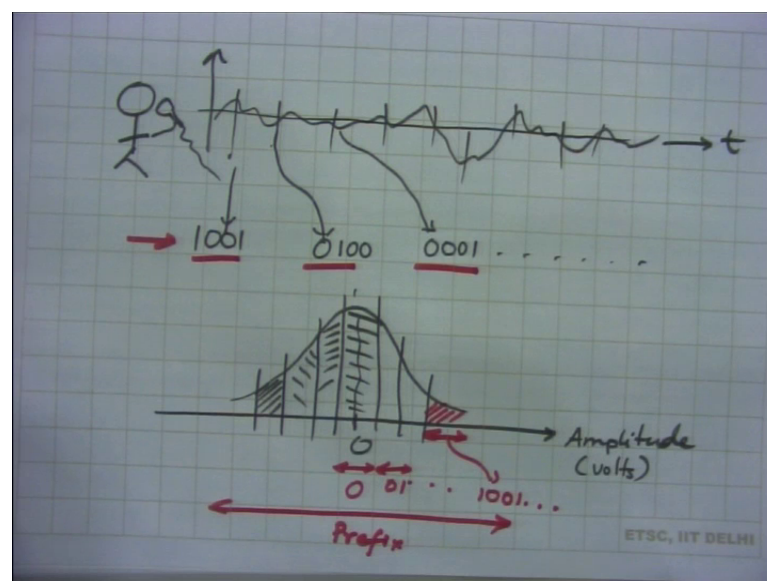
- The **Efficiency** of a prefix code is defined as
$$\eta = \frac{H(x)}{R}$$
- It is clear from the source coding theorem that the efficiency of a prefix code $\eta \leq 1$.
- Efficient representation of symbols leads to **compression** of data.
- Source coding is primarily used for compression of data (speech, text, image, video etc.).

 Indian Institute of Technology, Delhi30Ranjan Bose
Department of Electrical Engineering

Now let us talk about the efficiency of a prefix code; clearly we cannot go better than H of x . So, efficiency is a ratio which is less than or equal to 1 and η is given as H of X divided by R bar. So, it is clear from source coding theorem that η is less than or equal to 1 and it kind of represents the compression of data which is possible; this is routinely used for compressing speech text image audio video and so and so forth.

So, how do we do for example, speech compression? So, let us look at it.

(Refer Slide Time: 35:57)



So, we have a person who is speaking on a microphone and we have a waveform hello how are you. And so, you first have to digitize it for that we sample and whatever the values are there we quantize and these values need to be quantized in terms of 1001, 0100, 0001 and so on and so forth. Suppose I use 4 bits per sample, but how do we compress? Well, to compress we have to find out is there anything which is more frequently occurring is a; we less frequently occurring. So, we observe the waveform in time and we look at the amplitude and we are curious because it is random why do not we form a histogram of amplitude?

So, if we see that it jumps above 0 and below 0 and this is my 0 and this amplitude could be in volts. And we can possibly have a distribution and this distribution tells me that for this particular case my friend sticks normally around the 0 and once in a while screams. And therefore, the amplitude goes up, but he is a generally soft spoken person.

So, it is a high probability that my signal should lie close to 0 as opposed to having very high or very negative voltages, but it is a continuous random variable. So, we partition it and we found find the area under this curves which will tell us how much is the probability of occurrence. As is clear from this diagram that the probability of occurrences for different segments are different; so it is clear that low amplitude signals are more probable than high amplitude signals.

So, we should use fewer number of bits to represent low amplitude signals and so, and so, forth and the rarely occurring values should probably be represented by more number of bits. In fact, I can do a prefix code for different amplitude levels and thereby get compression. So, what we did earlier was not the best way because whether it was a high amplitude value or a low amplitude sample; we were assigning equal number of bits unfair, we can do it better by observing what is more frequent and what is less frequent.

We can do the same for text, we already looked at an example of your English alphabet with 8 characters, and we can also do it for video, audio, speech, text whatever you want.

(Refer Slide Time: 40:00)

Information Theory, Coding and Cryptography


Example

- Consider a source X which generates four symbols with probabilities $P(x_1) = 0.5$, $P(x_2) = 0.3$, $P(x_3) = 0.1$ and $P(x_4) = 0.1$.
- The entropy of this source is

$$H(X) = - \sum_{k=1}^4 P(x_k) \log_2 P(x_k) = 1.685 \text{ bits.}$$

- Suppose we use the prefix code $\{0, 10, 110, 111\}$
- Then the average codeword length, \bar{R} , is given by

$$\bar{R} = \sum_{k=1}^4 n_k P(x_k) = 1(0.5) + 2(0.3) + 3(0.1) + 3(0.1) = 1.700 \text{ bits.}$$

 Indian Institute of Technology, Delhi 31 Ranjan Bose
Department of Electrical Engineering

So, let us look at an example consider a source x which only generates 4 symbols Now was it a practical source well this is a source which generates 4 voltages. So, it could be a practical system, but these symbols are not equiprobable. So, I put $P(x_1)$ not visible equal to 0.5 second symbol occurs with probability 0.3, 0.1 and 0.1.

So, first of all we would like to see how what is the best we can do H of X plug in the values I multiply it out and I get 1.685 bits this is the entropy of this source. Now 4 symbols we have already done the binary tree; example we have 4 codewords which are forming the prefix code 0, 10, 110 and 111. So, what is the average codeword length \bar{R} we plug in $n_k P(x_k)$ summation of k equal to 1 to 4 and we get 1.7 bits. As expected \bar{R} is greater than 1.68 bits right, but not too far as we already know that my \bar{R} will be pretty close to H of X .

(Refer Slide Time: 41:35)


Information Theory, Coding and Cryptography

Example

- The efficiency of this code is
$$\eta = (1.685/1.700) = 0.9912.$$
- Had the source symbol probabilities been $P(x_k) = 2^{-n_k}$
i.e.,
 $P(x_1) = 2^{-1} = 0.5,$ $P(x_2) = 2^{-2} = 0.25,$
 $P(x_3) = 2^{-3} = 0.125$ and $P(x_4) = 2^{-3} = 0.125,$ \triangleright

The average codeword length, $\bar{R} = 1.750$ bits $= H(X).$

In this case, $\eta = 1.$

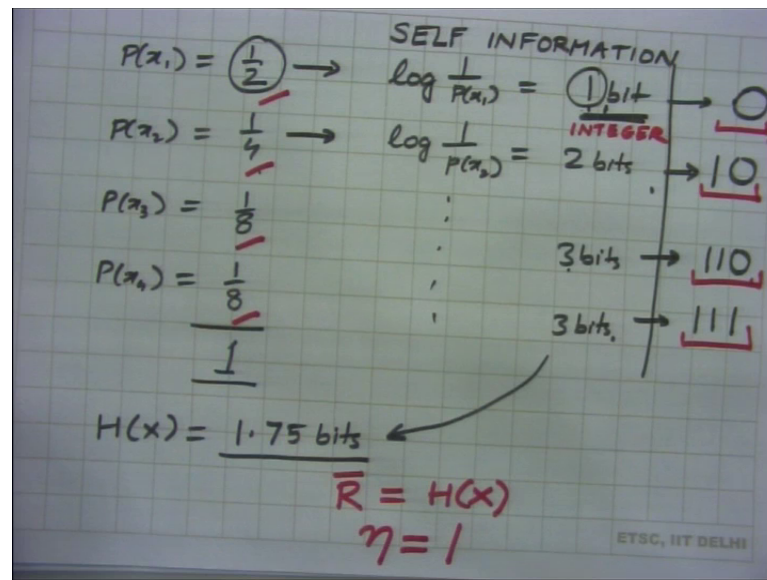
 Indian Institute of Technology, Delhi 32 Ranjan Bose
Department of Electrical Engineering

If we talk about the efficiency of this code we calculate this ratio 1.685 over 1.7 and pretty good 0.9912.

Now, suppose my symbols were different suppose instead of 0.5, 0.3, 0.1 and 0.1 I tweaked them and it becomes 0.5, 0.25, 0.125 and 0.125 you add them they add up to 1. And this time I plug in the value into my \bar{R} what is \bar{R} ? $\sum_{k=1}^N P(x_k) \log_2 P(x_k)$ over k I get the value 1.750. If I plug in these changed probabilities into my expression for H of X nothing but $-\sum p \log p$ I am surprised to see that \bar{R} is equal to 1.75 bits is equal to H of X . We have achieved the lower bound we have achieved the efficiency equal to 1.

So, what is strange about this condition? Why were we able to achieve this? Well it is easy to understand what is happening look at these probabilities $P(x_1) = 1/2$, $P(x_2) = 1/4$, $P(x_3) = 1/8$, $P(x_4) = 1/8$ and they add up to 1.

(Refer Slide Time: 43:17)



Now if we look at the average self information of this entire code which is H of X right you get 1.75 bits, but what is this average? Self information made out of it is made of self information.

So, let us look at the self information because x 1 occurs with probability 1 by 2, but what is the self information, how much respect does it command in terms of the bits? Well, it is log 1 over P x 1 equal to 1 bit. So, if you look at this guy you will get log 1 over P x 2 equal to right and so and so, forth. So, we are at 3 bits and 3 bits; now this is the information contained in these occurrences, but they do not occur with equal probabilities they occur with 1 by 2. So, if you multiply 1 by 2 into this plus 1 by 4 into this 1 by 8 into this 1 by 8 into this you get exactly this quantity.

Now, the good news is that I will use the more likely event probability 1 by 2 to be represented with fewer number of bits and since this is an integer I can have a 1-bit representation for this. So, as we had in the previous case we assign this as a 0; most frequently occurring minimum number of bits how many? Well self information is 1-bit I have given it 1-bit what about this guy? Well its self information is 2 bits; so, I give it 1 0 I give this guy 1 1 0 and 1 1 1.

. So, the only reason I could achieve the lower bound is because the self information is an integer. And I have only the luxury to allocate a full bit or 2 bits or 3 bits 2 and a certain outcome. I cannot if this had come out to be because of its probability 0.75, there

is no way I can offer 0.75 bit long codeword; the codeword length has to be an integer 1 2 3 3 and so and so forth.

And therefore, because of these very unique choices of probabilities which have the self information being integers; I have been able to achieve the lower bound of which R bar is equal to H of X and efficiency equal to 1 right. So, this is important to see where we are; so in most of the cases we will be far away from H of X .

(Refer Slide Time: 47:20)

Information Theory, Coding and Cryptography

Summary

- Variable Length Codes
- Kraft's Inequality
- Source Coding Theorem
- Efficiency of a code

Indian Institute of Technology, Delhi 33 Ranjan Bose
Department of Electrical Engineering

So, now we kind of summarize what we have learnt today; we started off with variable length codes because that said the premise for the source coding theorem. We looked at 2 different variable length code and we saw that one of them was uniquely decodable, the other one was not uniquely decodable. Then we looked at Kraft's inequality followed by the source coding theorem, which tells us the absolute minimum we can go for compression, and finally we looked at the efficiency of code.

So, that brings us to the end of this module.