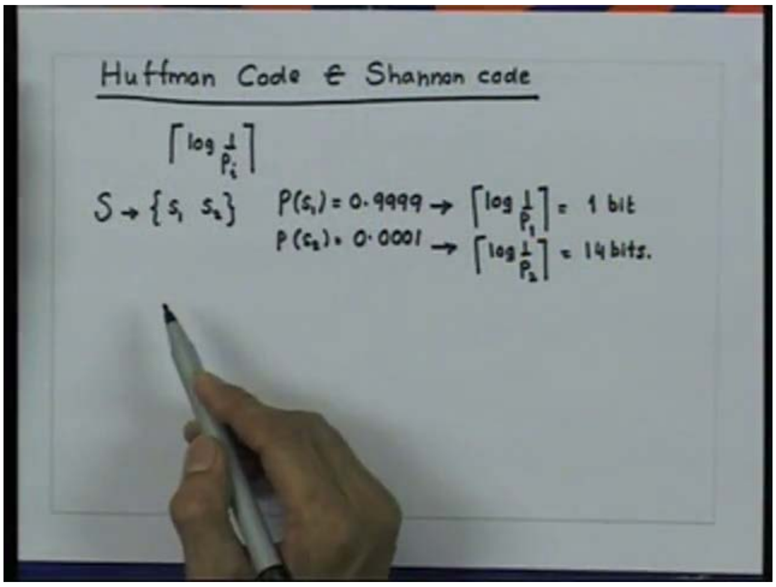
Information Theory and Coding
Prof. S. N. Merchant
Department of Electrical Engineering
Indian Institute of Technology, Bombay

Lecture - 13
Competitive Optimality of the Shannon Code

So, far we have studied three coding strategies. These are Shannon coding, Fano coding and Huffman coding. We have proved that Huffman code is optimal in the sense; that the expected or average code length for the Huffman code is minimum. Now, let us discuss a bit more about the code word length of Huffman code and Shannon code for some specific source alphabet.

(Refer Slide Time: 01:30)



Huffman Code & Shannon code

$$\left\lceil \log \frac{1}{P_i} \right\rceil$$

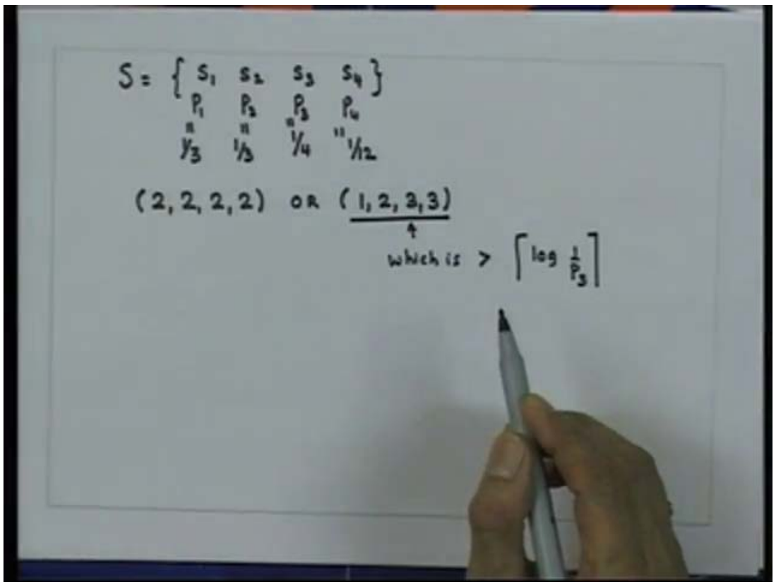
$$S \rightarrow \{s_1\ s_2\} \quad P(s_1) = 0.9999 \rightarrow \left\lceil \log \frac{1}{P_1} \right\rceil = 1 \text{ bit}$$

$$P(s_2) = 0.0001 \rightarrow \left\lceil \log \frac{1}{P_2} \right\rceil = 14 \text{ bits.}$$

Huffman code and Shannon code, we will look at this two codes and look at their word length. Now, using code word lengths of log 1 by P I, which is called Shannon coding. This may be sometime much worse than the optimal code for some particular symbol. Let us take an example that I have a source a. Binary source consisting of two symbols S 1 and S 2 with the probabilities given as probability of S 1 is equal to 0.9999 and probability of S 2 given as 0.0001.

Now, if we use the Shannon coding strategy, then we can calculate the number of bits which we should allot to the source symbol S 1 as log of upper flow of log of 1 by P 1 and that will come out to be 1 bit. For this the upper flow of log of 1 by P 2 will be a code to 14 bits. Optimal code word length is obviously 1 bit for both the symbols S 1 and S 2, hence the code for the infrequent symbol is much longer in the Shannon code than in the optimal code like Huffman code. Now, is it true that, code word length for an optimal code are always less than upper flow of log 1 by P i? Let us answer this question with an example of an illustration.

(Refer Slide Time: 03:58)



$$S = \begin{Bmatrix} s_1 & s_2 & s_3 & s_4 \\ P_1 & P_2 & P_3 & P_4 \\ = & = & = & = \\ 1/3 & 1/3 & 1/4 & 1/12 \end{Bmatrix}$$

$(2,2,2,2)$ OR $\underline{(1,2,3,3)}$

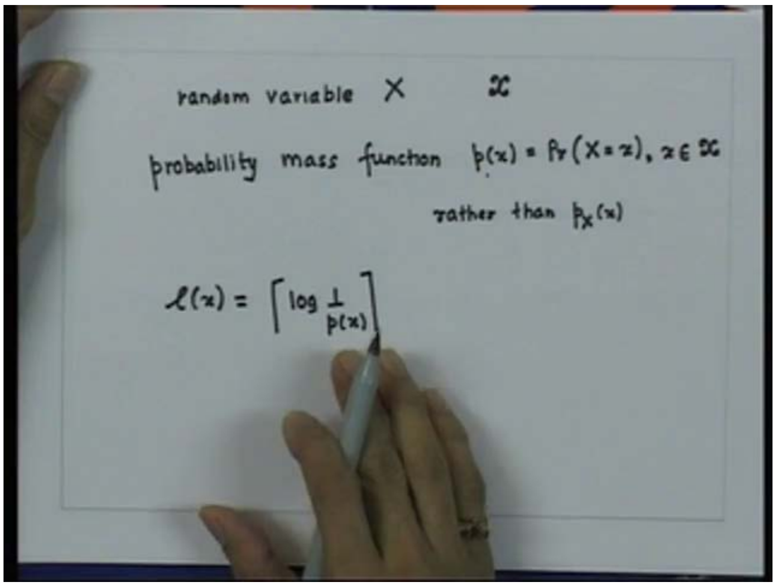
$\uparrow$ which is $> \left\lceil \log \frac{1}{P_3} \right\rceil$

Let us assume that I have a source consisting of four symbols with the probabilities given as one-third probability of a S 2 equal to one-third probability of S 3 equal to one-fourth and probability of S 4 equal to 1 by 12. Now, for this source we can design an Huffman code and the code word length, which will result from the design of the Huffman code will be 2 2 2 2 or 1 2 3 3 depending on where one puts the merge.

Probabilities in the process of Huffman coding to get this, we follow the exactly same procedure, which we had discussed in the last class. Now, both this Huffman codes achieve the same expected code word length in the second code; that is this the third symbol has length 3, which is greater than log of 1 by P 3. Thus the code word length for a Shannon code which would be decided by this could be less than the code word length of the corresponding symbol of an optimal Huffman code. Huffman code is optimal in that, it has a minimum expected length, but what does that say about its performance on any particular sequence?

For example, is it always better than any other code for all sequencing, obviously not. Since, there are codes, which assign short code words to infrequent source symbols. Now, such code will be better than the Huffman code on those source symbol is t would be interesting to relate the code word lengths. At least in probabilistic sense of say Huffman code or Shannon code to the code word length of some other uniquely decodable code. Now, for notational convenience and this in turn will result in the simplification of understanding of the concept to follow. We will follow the following framework, we will denote the output of a source as a random variable.

(Refer Slide Time: 07:24)


random variable X $\mathcal{X}$

probability mass function $p(x) = \Pr(X = x), x \in \mathcal{X}$

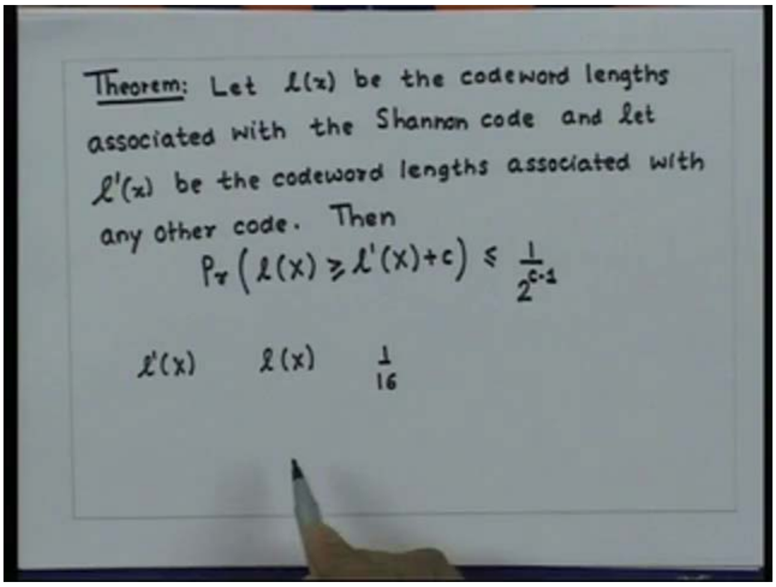
rather than $p_X(x)$

$$l(x) = \left\lceil \log \frac{1}{p(x)} \right\rceil$$

So, the output of a source will be denoted by a random variable x with the source alphabet given by this. We will denote the probability mass function probability mass function p x; that is equal to probability of the random variable x equal to x where x belongs to the alphabet. We will also denote this probability mass function p x by we are denoting this by rather than by this notation just for convenience. So, if you want to denote the probability mass function for random variable x, we should denote it by this term. But just for the sake of convenience, we will denote it by a simplification simplified form as just p x.

Now, dealing with Huffman code is difficult, since there is no explicit explanation for the code word length instead. We will consider the Shannon code because for the Shannon code word lengths i l x can be specified as equal to log of 1 by p x such explicit explanation for the code word length of Huffman length is not possible. So, it will be difficult for us to consider Huffman code, so we consider the Shannon code and for the Shannon code we have a following important theorem.

(Refer Slide Time: 09:51)



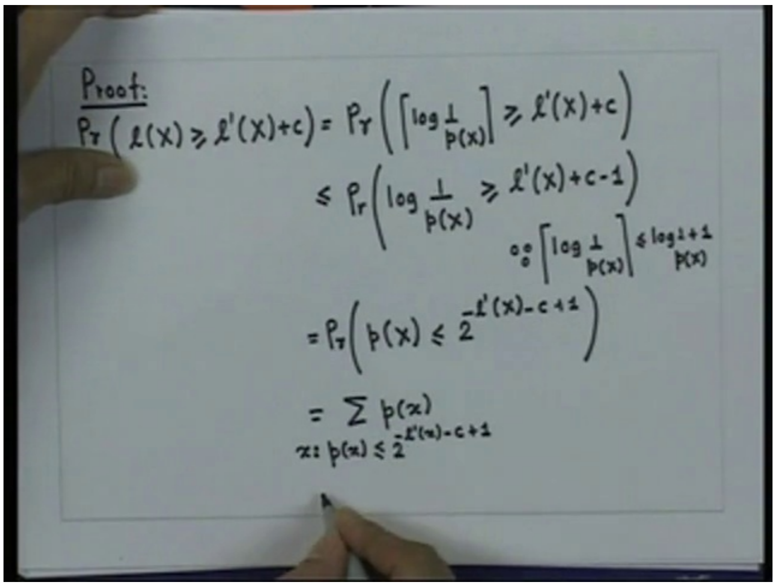
Theorem: Let $\ell(x)$ be the codeword lengths associated with the Shannon code and let $\ell'(x)$ be the codeword lengths associated with any other code. Then

$$P_r\left(\ell(x) \geq \ell'(x)+c\right) \leq \frac{1}{2^{c-1}}$$

$\ell'(x)$ $\ell(x)$ $\frac{1}{16}$

Let l x be the code word length associated with the Shannon code and let l dash x be the code word lengths associated with any other code. Then the theorem says that probability of l x is greater than equal to plus a constant c is less than equal to 1 by 2 c minus 1. what it means is that the probability that l dash x is say 5 or more bits shorter. Then l x is less than 1 by 16, so the interpretation of this is given as explained. Now, let us look into the proof of this.

(Refer Slide Time: 12:18)



<u>Proof:</u>

$$\Pr\left(l(X) \geq l'(X)+c\right) = \Pr\left(\left\lceil \log \frac{1}{p(X)} \right\rceil \geq l'(X)+c\right)$$

$$\leq \Pr\left(\log \frac{1}{p(X)} \geq l'(X)+c-1\right)$$

$$\because \left\lceil \log \frac{1}{p(X)} \right\rceil \leq \log \frac{1}{p(X)} + 1$$

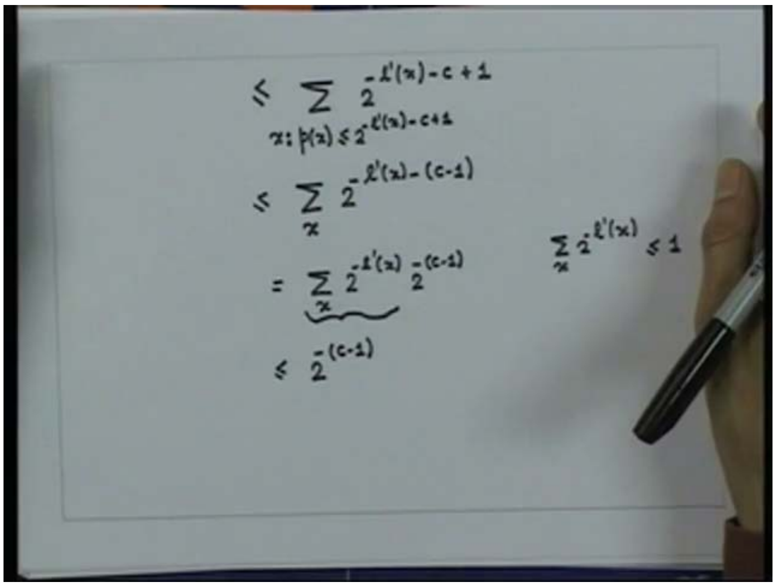
$$= \Pr\left(p(X) \leq 2^{-l'(X)-c+1}\right)$$

$$= \sum_{x:\, p(x) \leq 2^{-l'(x)-c+1}} p(x)$$

So, probability of random variable greater than l dash x plus C is nothing but probability of log of 1 by P greater than equal to this, this implies this, because the length is given by the full flow 1 by p x for the Shannon code. Now, this quantity is less than equal to probability of log of 1 by p of x greater than equal to length dash l dash x plus c minus 1. I can write this expression because log of is less than equal to log of 1 by p x plus 1. Because of this I can write this expression.

Now, this interpretation in terms of probability is nothing but probability of x less than or equal to 2 minus l dash x minus c plus 1. Now, probability of this random variable is nothing but a summation of the individual probabilities, where I have to sum up for all x which satisfies p of x less than or equal to 2 minus c plus 1. So, this is equivalent to summing up the probabilities of x where x is all those x for which probability of x is less than equal to this quantity. Now, I am summing this up for all p x less than this quantity. So, what it implies that this is this quantity itself is less than or equal to summation 2 minus l dash x minus c plus 1 for this is the upper bound of p x.

(Refer Slide Time: 16:08)



$$\leq \sum_{x: p(x) \leq 2^{-l'(x)-c+1}} 2^{-l'(x)-c+1}$$

$$\leq \sum_{x} 2^{-l'(x)-(c-1)}$$

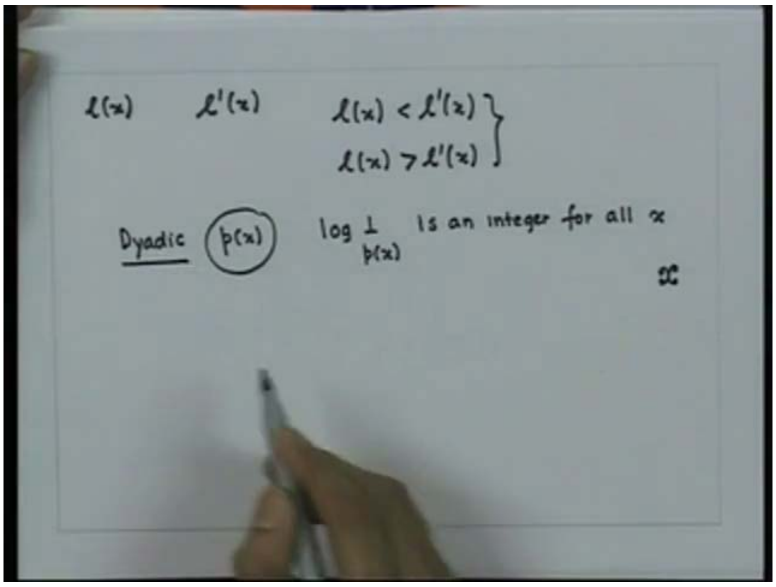
$$= \underbrace{\sum_{x} 2^{-l'(x)}}\, 2^{-(c-1)}$$

$$\leq 2^{-(c-1)}$$

$$\sum_{x} 2^{-l'(x)} \leq 1$$

So, that is why it becomes less than or equal to… Now, this is obviously less than where i sum up for all x. So, 2 minus l dash x minus c minus 1, so this is equal to… Now, this quantity is the craft inequality, so summation of 2 minus l dash x. If I assume that the code which I am considering is a uniquely decodable, then this should satisfy Macmillan theory of inequality. This is less than equal to 1, so what it implies? That this quantity is less than equal to 2 minus c minus 1, so we have proved that no other code can do much better than the Shannon code most of the time. So, it is a very important result. Now, pertaining to a channel four, there is another important theorem, which we will have a look at it.

(Refer Slide Time: 18:38)



$l(x)$ $\quad l'(x)$ $\quad \left.\begin{array}{l} l(x) < l'(x) \\ l(x) > l'(x) \end{array}\right\}$

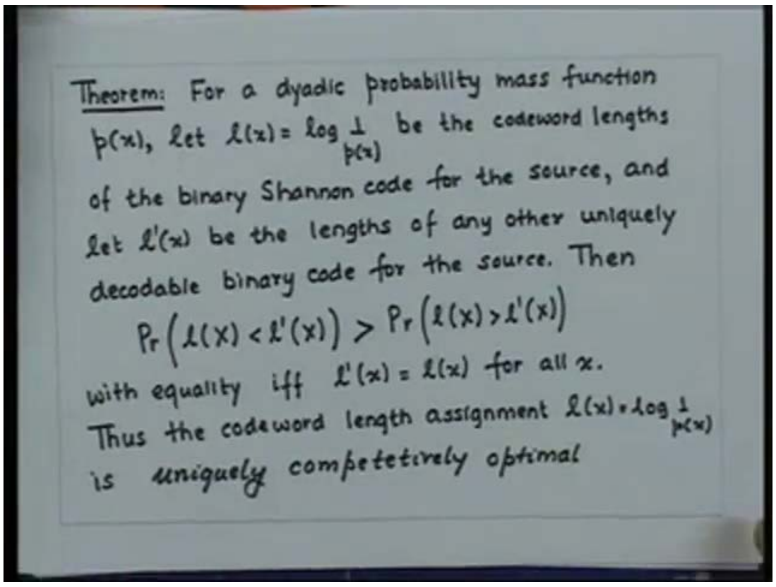
Dyadic $\boxed{p(x)}$ $\log \frac{1}{p(x)}$ is an integer for all x

$\mathcal{X}$

So, suppose I have the code word length l x for the Shannon code and l dash x as the code word length for any other uniquely decodable code. Then what one would like to ensure is that l x is less than l dash x, more often than l x is greater than l dash x. So, this is something which is desirable that I want l x, which pertains to the Shannon code is less than l dash x, which pertains to any other uniquely decodable code. This should happen more often than this, so that is probability of this should be more than this.

So, let us try to prove that, but before we try to prove that. Let me define, what is the dyadic probability mass function p x? Now, probability mass function p x is dyadic, if log of 1 by p x is an integer for all x. x comes from is a discrete and it comes from the alphabet this x. So, if log of 1 by p x is an integer for all x, then probability mass function which is given by p x is known as dyadic.

(Refer Slide Time: 20:55)



<u>Theorem:</u> For a dyadic probability mass function $p(x)$, let $l(x) = \log \frac{1}{p(x)}$ be the codeword lengths of the binary Shannon code for the source, and let $l'(x)$ be the lengths of any other uniquely decodable binary code for the source. Then

$$\Pr\left(l(X) < l'(X)\right) > \Pr\left(l(X) > l'(X)\right)$$

with equality iff $l'(x) = l(x)$ for all x.

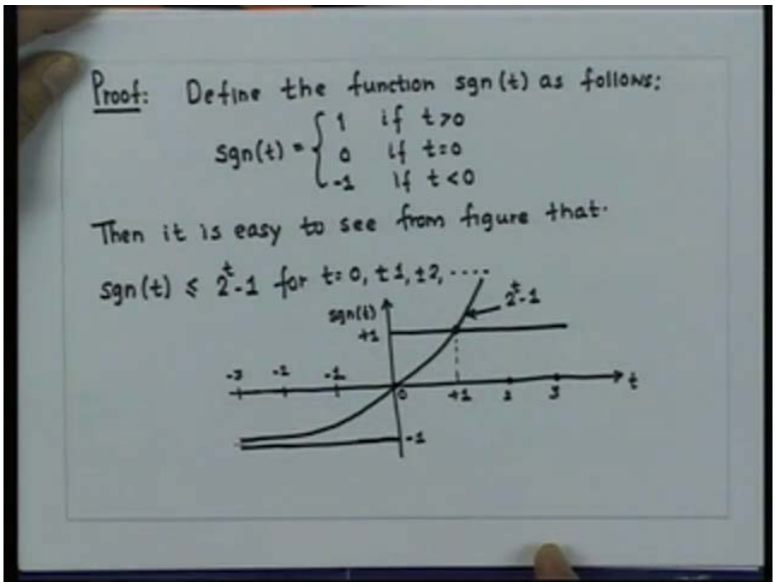
Thus the codeword length assignment $l(x) = \log \frac{1}{p(x)}$ is *uniquely competetively optimal*

Now, with this definition let me study another important theorem. The theorem says that for a dyadic probability mass function p x. Let l x be equal to log 1 by p x be the code word length of the binary Shannon code for the source and let l dash x be the lengths the code word length of any other uniquely decodable binary code for the same source. Then the theorem says that probability of l x being less than l dash x is greater than probability of l x being greater than l dash x.

This is with equality if and only if l dash x is equal to l x for all x. Now, we can say thus the code word length assignment of the form l x equal to log of 1 by p x; that is the Shannon's strategy. If uniquely competitively optimal, so we have proved that Huffman code is optimal in the sense that the average expected code length is minimal. But now, what we are going to prove is that, Shannon code is uniquely competitively optimal and by that we mean?

That probability of l x being less than l dash x the probability of code word length for Shannon code being less than for any other uniquely decodable code, is always greater than probability of code word length of Shannon code being greater than the code. Word length of a any other uniquely decodable code a very important result though this result. We are trying to prove it for the dyadic probability mass function. Then we will try to relax this condition and see what happens so the proof follows as follows.

(Refer Slide Time: 25:42)



Proof: Define the function sgn(t) as follows:

$$\mathrm{sgn}(t) = \begin{cases} 1 & \text{if } t > 0 \\ 0 & \text{if } t = 0 \\ -1 & \text{if } t < 0 \end{cases}$$

Then it is easy to see from figure that

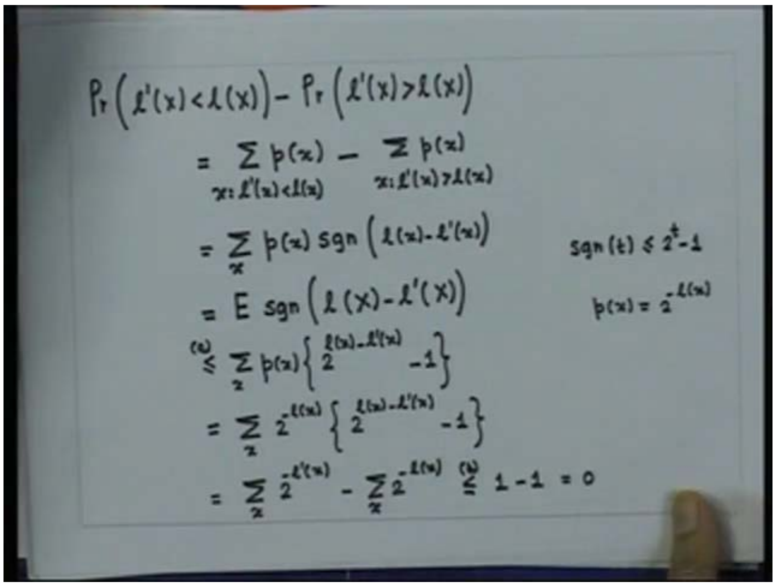
$\mathrm{sgn}(t) \leq 2^t - 1$ for $t = 0, \pm 1, \pm 2, \ldots$

sgn(t)
+1
2^t-1
-3
-2
-1
0
+1
2
3
t
-1

Now, before we have a look at the proof, let me define one function that is the signum function. Define the function signum t as follows signum t is defined as, it is equal to 1 is tis greater than 1 is equal to 0. If t is equal to 0 and it is equal to minus 1, if t is less than 0. Then based on this definition, it is easy to see from the following figure that signum t will be always be less than or equal to 2 raise to t minus 1 for t equal to 0 plus minus 1 plus minus 2. So, if we draw the plot for these two functions, they will appear something like this t and i plot my function.

So, this is my signum t and this function is 2 raise to t minus 1. So, from this is a plot for 2 raise to t minus 1 and this is a plot for the function signum t form this plot. It is very clear that signum t is always less than or equal to 2 raise to t minus 1, but very important to note that this is valid for t equal to 0 plus minus 1 plus minus 2. So, it is for all the integer values this is true and 2 t minus 1 when t tends to negative is bounded by minus 1. So, this result is always valid. Now, based on this result, we will try to prove the theorem for this is not valid for all t, but valid for integer values of t.

(Refer Slide Time: 29:26)



$$
\begin{aligned}
&P_r\left(l'(x) < l(x)\right) - P_r\left(l'(x) > l(x)\right) \\
&= \sum_{x: l'(x) < l(x)} p(x) - \sum_{x: l'(x) > l(x)} p(x) \\
&= \sum_{x} p(x)\, \mathrm{sgn}\left(l(x) - l'(x)\right) \\
&= E\, \mathrm{sgn}\left(l(X) - l'(X)\right) \\
&\overset{(a)}{\leq} \sum_{x} p(x)\left\{2^{l(x)-l'(x)} - 1\right\} \\
&= \sum_{x} 2^{-l(x)}\left\{2^{l(x)-l'(x)} - 1\right\} \\
&= \sum_{x} 2^{-l'(x)} - \sum_{x} 2^{-l(x)} \overset{(b)}{\leq} 1 - 1 = 0
\end{aligned}
$$

$\mathrm{sgn}(t) \leq 2^{t} - 1$

$p(x) = 2^{-l(x)}$

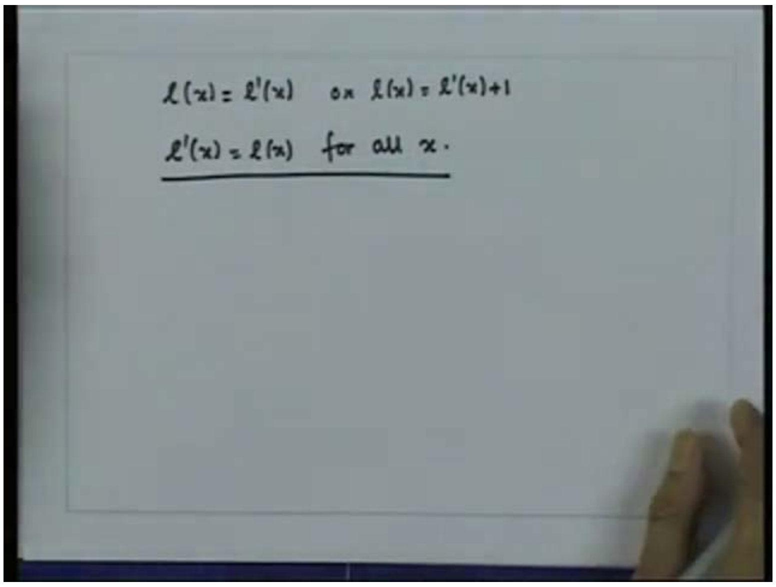
So, based on this we can now write probability of l dash x being less than l x minus probability of l dash x being less than l x minus probability of l dash x being greater than l x. This I can write this probability, I can write as probability of x where my I consider only those x for which l dash x is less than l x, the probability of the random variable. So, by the definition it is this and this is nothing but the probability of the symbol for which l dash x will be greater than l x. Now, this itself very it is not very difficult that this I can write as summation of p x this is signum of l x minus l dash x for overall x. It is simple from here to here.

Now, this by definition is nothing but expectation of signum function l x minus l dash x. This follows from the definition of expectation now because we have seen that signum of t is always less than equal to 2 raise to t minus 1 for t equal to 0 plus minus 1 plus minus 2. Since, our difference between this two random variables are always integers, we can use this relationship to write as summation of x p x 2 of raise to l x minus l dash x minus 1. So, let me just denote this relationship out here by a we will come back to this little later. So, I denote this step out here by a. Now, based on this we have come from here to here.

Now, this I can simplify because my p x because a dyadic probability mass function. So, p x is nothing but two raise to minus l x. So, I can write this as summation of x 2 raise to minus l x minus 1 and this we can simplify as x 2 raise to minus l dash x minus 2 raise to minus l of x. Now, this quantity is nothing but the Macmillan's graph inequality. So, if your l dash x corresponds to the code word length of another uniquely decodable code, then it implies that this quantity has to be less than equal to 1.

So, I can say that this is equal to less than or equal to 1 and this is obviously 1, because a dyadic probability mass function. So, summation again is equal to 1. So, let me call this relationship as b. So, this quantity out here is going to be less than equal to 1, if I can write this as less than equal to 1 minus 1 and if equal to 0, so what we have shown? Now, that this is less than equal to 0, where this step a follows from this and b here. We have used the Macmillan graph inequality. Now, we have an equality in the bound at a and b.

(Refer Slide Time: 35:11)



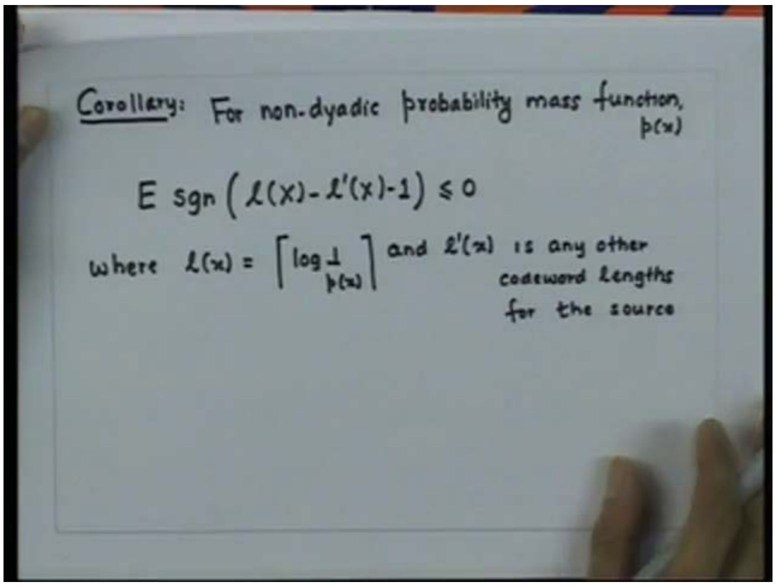
$l(x) = l'(x)$ or $l(x) = l'(x)+1$

$\underline{l'(x) = l(x) \text{ for all } x.}$

So, if we have equality in the bound for signum, which is occurred only for t equal to 0 or 1 in this case, then if p if you want to have a equality here, then what it means? That l x is equal to l dash x or l x is equal to l dash x plus 1. This is I can write because the signum function. If we look at the signum function, the equality holds for t equal to 0 and t equal to 1, so in our case here if I assume that the equality at this point, then t equal 0 and t equal to 1 corresponds to l x is equal to l dash x and or l x is equal to l dash x plus 1. Another equality is if you want the equality then this at point b, this should be equal to 1, which implies that the graph inequality is satisfied and combining these two facts we will get l dash x is equal to l x for all x.

So, this we get from equality a, I mean when we assume the equality a when we assume this condition that t is equal to 0 or 1 at this point. Here we assume that this also satisfies the equality of Macmillan graph equality. Then from both this condition, we get l dash x is equal to l x for all x. So, we have proved the theorem, so we have proved that this probability is always greater than this for a Shannon code. So, after having proved that the Shannon code is uniquely competitively optimal, so after having proved that the Shannon code is uniquely competitively optimal for a dyadic probability mass function.

(Refer Slide Time: 37:37)



Corollary: For non-dyadic probability mass function, $p(x)$

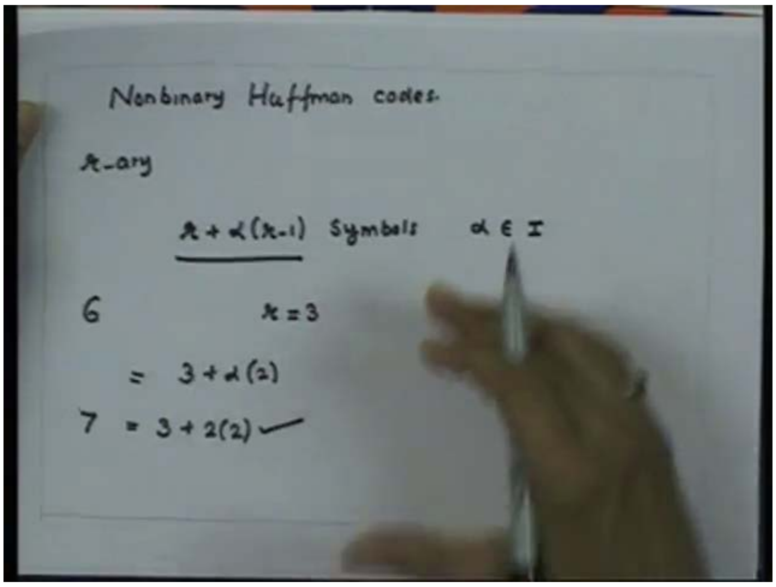
$$E \operatorname{sgn}\left(l(X) - l'(X) - 1\right) \leq 0$$

where $l(x) = \left\lceil \log \frac{1}{p(x)} \right\rceil$ and $l'(x)$ is any other codeword lengths for the source

For non dyadic probability mass function there is a corollary. I just take the corollary for the sake of completion without the proof. But the proof is followed along the same line as the proof, which we saw for the dyadic probability mass function for the corollary. For the non-dyadic probability mass function would be for non-dyadic probability mass function, expectation of signum l x minus l dash x minus 1 is less than or equal to 0, where l x is upper flow of log of 1 by p x and l dash x corresponds to the code word length for the source, but this is another uniquely decodable code. So, l dash x is any other code word lengths the uniquely decodable code for the source.

So, a important result, what it says that on the average the Shannon code when provided the code word length which does not differ from the code word lengths of any other uniquely decodable code by 1. So, important relationship after having studied this, not let us go back to the Huffman code. Now, we had looked into the procedure for the design of a Huffman code where our code alphabet was a binary in nature. Now, let us look at the procedure to extend the binary a Huffman coding to a non-binary Huffman coding strategy. So, in the non-binary Huffman coding, what we mean is that the code alphabet is size is not 2, but greater than 2. So, the binary Huffman coding procedure can be easily extended to the non binary case where the code elements form an r array alphabet.

(Refer Slide Time: 40:54)



Nonbinary Huffman codes.

r-ary

$\underline{r + \alpha(r-1)}$ symbols $\alpha \in I$

6 $r = 3$

$= 3 + \alpha(2)$

$7 = 3 + 2(2)$ ✓

So, let us study non binary Huffman code, because that we had obtained the huffman algorithm based on the two observations; that first that the symbols that occur more frequently or have a higher probability of occurrence. We will get from 6 letters alphabet. We will have a shorter code word than symbols that occur less frequently and the second observation was that the two symbols that occur least frequently will have the same length. We added an additional requirement that the two symbols with the lowest probability differ only in the last portion. Now, we can obtain a non binary Huffman code in almost exactly the same way.

The obvious thing to do would be to modify the second observation, which would mean… Now, that the r symbols, r denotes the size of the code alphabets. So, the r symbols which occur least frequently will have the same length that is the modification of the second observation. We will also modify the additional requirement to reduce the r symbols with the lowest probability will differ only in the last position. However, we run into a small problem with this approach consider design of a ternary Huffman code for a third with 6 letter alphabet.

Now, is we use the rule described. Now, then in this case, we would do is we would first combine the 3 letters with the lowest probability into a composite letter. So, that we would be doing at the first stage, so after the first reduction, we will move over to 4 letter alphabet. Now, once we move over to the 4 letter alphabet and if we follow same procedure of combining 3 letters into a composite letter, then at the end of the second stage of reduction, I would be getting 2 letters. So, finally, I have a 2 letters reduced alphabet and I have three symbols to be allotted, so this is the one strategy which I could have followed.

Another strategy, which I could have followed was that to start initially, I could have combined 2 letters, so when I combine 2 letters then at the first stage of my reduction, I would get 5 letters reduced alphabet. Now, once I get 5 letter reduced alphabet, then I can combine again 3 letters into a composite letter. When I do that at the second reduction, I would get 3 letters. I have three symbols from the code alphabets to be allotted to the three reduced alphabet, so no problem. Now, I could have followed another strategy, another strategy could have been that initially I combine 3 letters. So, at the first stage of my reduction I get 4 letters.

At this stage instead of combining 3 letters, I would combine 2 letters. Finally, I would get 3 letters. Now, there are three alternatives, which I have suggested. So, the question is, which of these alternatives should be followed? Now, recall that the symbol with the lowest probability will always have the longest code word. Furthermore, all the symbols that we combine together into a composite symbol will have code words of the same length. What this mean? That all the letters we combine together at the very first stage will have code word that have the same length. This code words will be the longest of all the code words, so this being the case if at some stage we have allowed to combine less than r symbols.

Then obviously the logical place to do this would be in the very first stage. So, in order to achieve this, we can add some dummy source symbols with probabilities 0. Then combine r symbols at a time at each stage of the reduction indirectly by this procedure. What we are doing? That the very first stage the symbols, we are combining are less than r, because the other symbols, which we added as a dummy symbol are having probability equal to 0. Why we do this is, because we can just like we prove for the binary Huffman code; that if we want the original tree to be optimal, then this necessary that the reduced tree.

We develop a code for the reduced tree also which is optimum optimal. Now, in order to do that, we have to always combine r symbols at a time. So, if wish to form an r array contact the optimal code that an any particular step in the Huffman coding algorithm. We shall combine the source symbols r at a time in order to form one symbol in the reduced source in the sequence. Now, we would like the last source in the sequence to have exactly r symbols. This will allow us to construct the trivial optimal code for this source by allotting each code symbol to the source symbol. The last source will have exactly r symbols, if and only if the original source has r plus alpha r minus one symbols, where alpha is an positive integer.

So, only the original symbols satisfy this relationship where r denotes the size of the code alphabet and alpha denotes the integer, if my original size of my source alphabet is of the form this. Then I can combine r symbols at a time at ease reduction of the source and this will provide me the optimal code, when I move backwards. Therefore, if the original source does not have this many number of symbols, we add dummy symbols to the source until this number is reached. Now, it is not very difficult to show this, take a simple example. Suppose, I start with three source symbols then three source symbols alpha is equal to 0 and it satisfies this condition.

Now, if I have my six source symbols, so if take let us take an example. I have six source symbols to start with and I want to do the coding using code alphabet of size r equal to 3. So, if you look at this expression we get 3 plus alpha 2, which is of this 6 is not of this form. So, what it means that to make of this form I should add some dummy variable. So, if add one dummy variable here that is say, make it 7, then 7 is of the form 3 plus 2 into 2. So, if add one dummy variable, which is has a probability of 0, then I get this relationship satisfied. This also tells me that if I have six source symbols, we try to understand the non binary Huffman coding procedure, which we have discussed today with the help of an example and that we will do in the next class.