

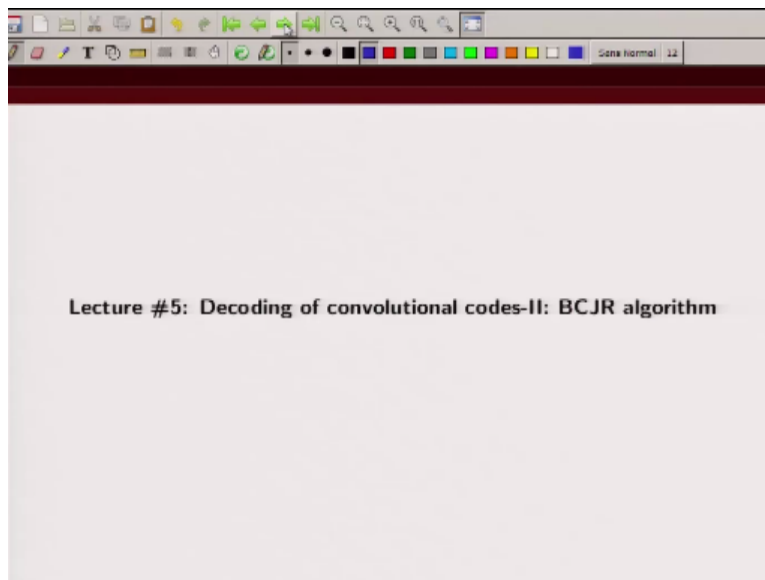
**Indian Institute on Technology Kanpur**  
**National Programme on Technology Enhanced Learning (NPTEL)**  
**Course Title**  
**Error Control Coding: An Introduction to Convolutional Codes**

**Lecture-5**  
**Decoding of Convolutional Codes-II: BCJR Algorithm**

by  
**Prof. Adrish Banerjee**  
**Dept. Electrical Engineering, IIT Kanpur**

Welcome to the course on error control coding, an introduction to convolutional code.

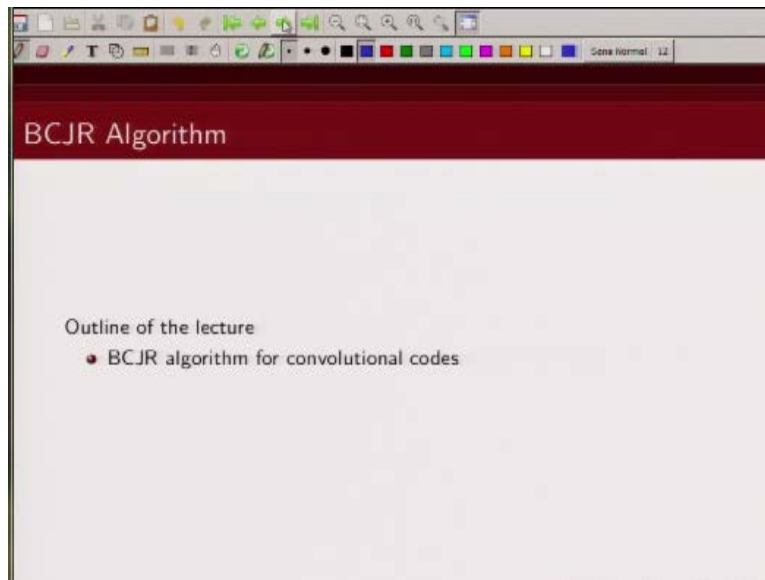
(Refer Slide Time: 00:23)



We are going to continue our discussion on decoding of convolutional codes. In the last class we talked about Viterbi decoding and if you recall Viterbi decoding is an efficient algorithm to compute a path to the trellis for convolutional code. Now it essentially finds out, Viterbi algorithm essentially finds out an estimate of the code word, because any path through the trellis of a convolutional code is basically a code word.

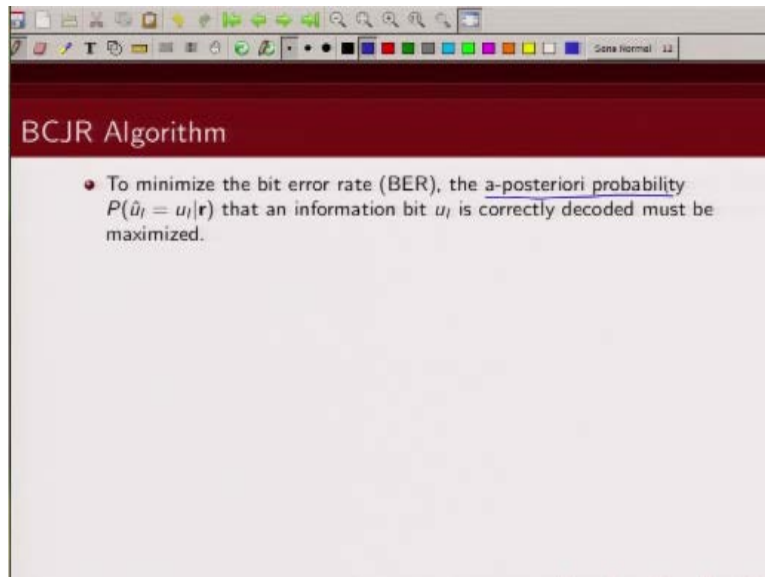
Now that not necessarily minimizes the bit error rate probability. In many applications we are interested to minimize the bit error rate.

(Refer Slide Time: 01:09)



So today we are going to talk about a decoding algorithm which is basically going to minimize bit error rate probability, symbol error rate probability.

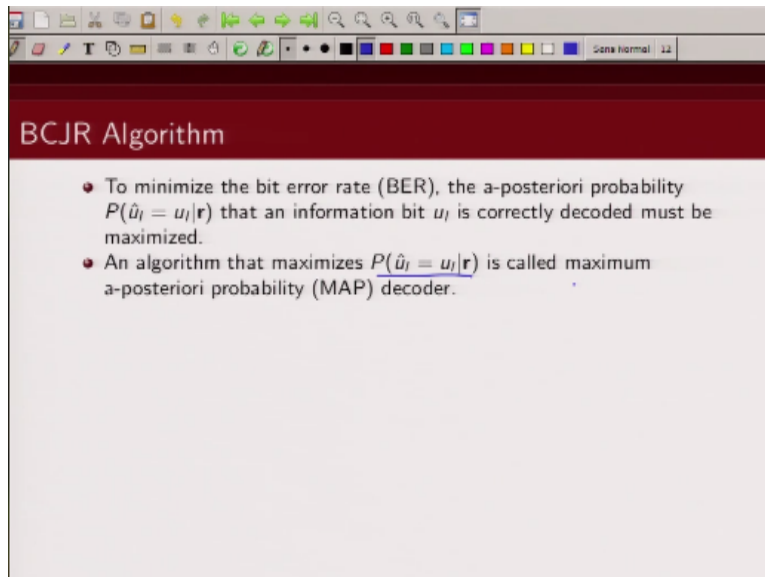
(Refer Slide Time: 01:19)



The image shows a screenshot of a presentation slide. At the top, there is a dark red header bar with the text "BCJR Algorithm" in white. Below the header, the slide content is on a light gray background. A single bullet point is visible, starting with a red dot. The text of the bullet point reads: "To minimize the bit error rate (BER), the a-posteriori probability  $P(\hat{u}_l = u_l | \mathbf{r})$  that an information bit  $u_l$  is correctly decoded must be maximized." The slide is displayed within a window that has a standard operating system toolbar at the top, including icons for file operations, editing, and navigation. The window title bar shows "Sona Normal 12".

So we are going to use a a-posteriori probability based algorithm to estimate our information sequence.

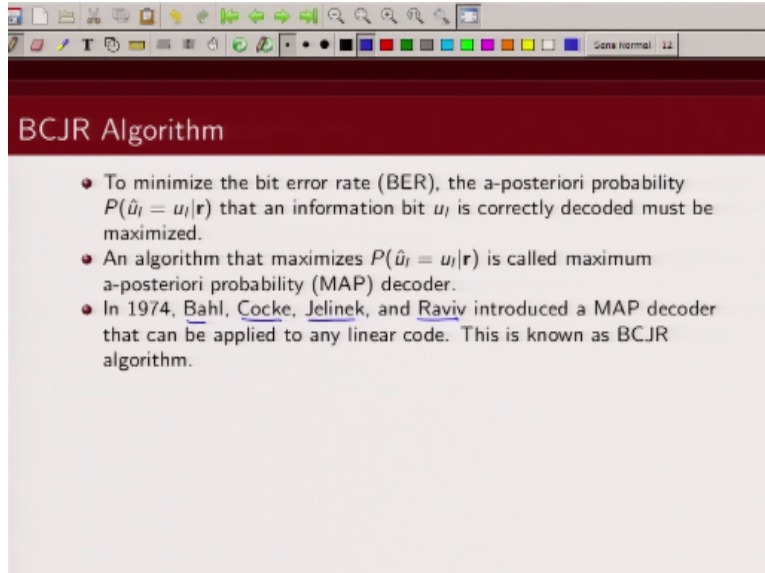
(Refer Slide Time: 01:31)



The image shows a screenshot of a presentation slide. The slide has a dark red header with the text "BCJR Algorithm" in white. Below the header, there is a light gray background with two bullet points. The first bullet point states: "To minimize the bit error rate (BER), the a-posteriori probability  $P(\hat{u}_l = u_l | \mathbf{r})$  that an information bit  $u_l$  is correctly decoded must be maximized." The second bullet point states: "An algorithm that maximizes  $P(\hat{u}_l = u_l | \mathbf{r})$  is called maximum a-posteriori probability (MAP) decoder." The slide is displayed in a window with a standard toolbar at the top.

And this algorithm which maximizes probability of  $\hat{u}$  given  $u$ , given the received sequence  $r$  is known as map decoder.

(Refer Slide Time: 01:47)



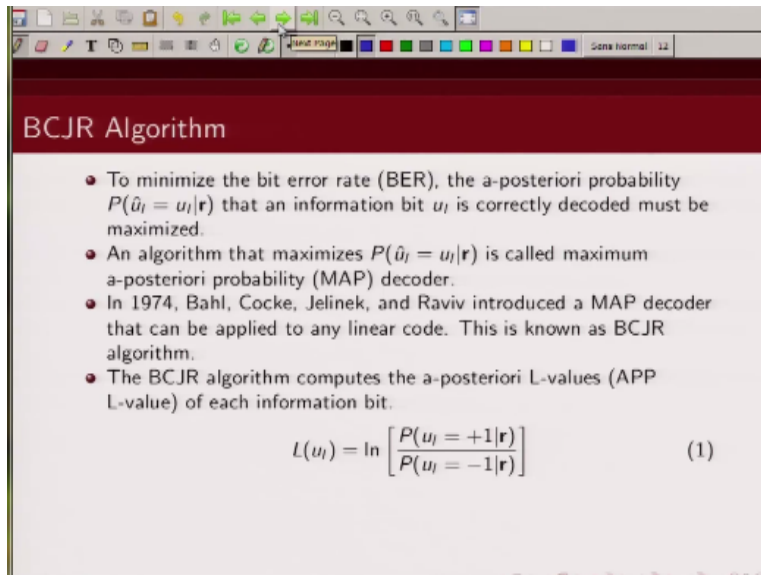
The image shows a screenshot of a presentation slide. The slide has a dark red header with the title "BCJR Algorithm" in white text. Below the header, there is a list of three bullet points. The first bullet point discusses minimizing the bit error rate (BER) by maximizing the a-posteriori probability  $P(\hat{u}_l = u_l | \mathbf{r})$ . The second bullet point states that an algorithm maximizing this probability is called a maximum a-posteriori probability (MAP) decoder. The third bullet point mentions that in 1974, Bahl, Cocke, Jelinek, and Raviv introduced a MAP decoder applicable to any linear code, known as the BCJR algorithm. The slide is displayed in a window with a standard toolbar at the top.

### BCJR Algorithm

- To minimize the bit error rate (BER), the a-posteriori probability  $P(\hat{u}_l = u_l | \mathbf{r})$  that an information bit  $u_l$  is correctly decoded must be maximized.
- An algorithm that maximizes  $P(\hat{u}_l = u_l | \mathbf{r})$  is called maximum a-posteriori probability (MAP) decoder.
- In 1974, Bahl, Cocke, Jelinek, and Raviv introduced a MAP decoder that can be applied to any linear code. This is known as BCJR algorithm.

Now this is known as – also known as BCJR algorithm named after these researchers who Bahl, Cocke, Jelinek and Raviv who introduced this algorithm in 1974. And this algorithm can be applied to any linear code, block code or convolutional code.

(Refer Slide Time: 02:08)



The image shows a screenshot of a presentation slide titled "BCJR Algorithm". The slide is displayed in a window with a standard toolbar at the top. The title "BCJR Algorithm" is in a dark red header. Below the title, there is a list of four bullet points explaining the algorithm's purpose and history. At the bottom of the slide, there is a mathematical equation for the L-value of an information bit.

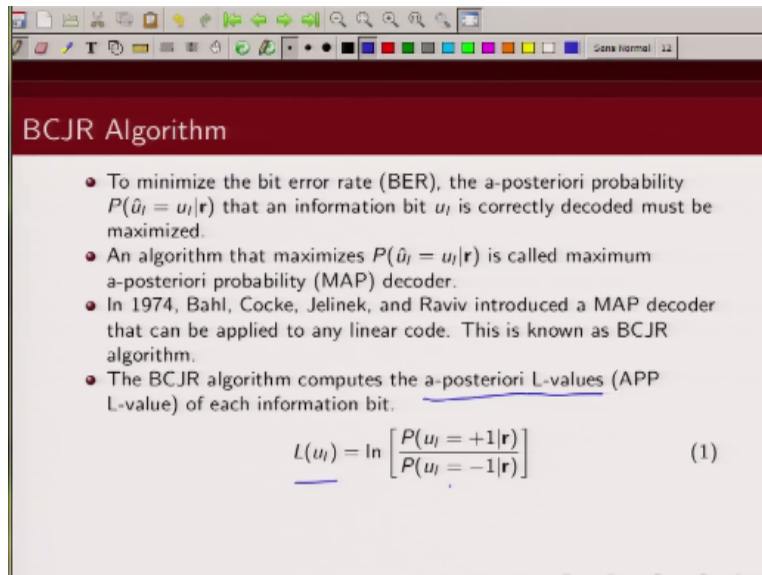
**BCJR Algorithm**

- To minimize the bit error rate (BER), the a-posteriori probability  $P(\hat{u}_i = u_i | \mathbf{r})$  that an information bit  $u_i$  is correctly decoded must be maximized.
- An algorithm that maximizes  $P(\hat{u}_i = u_i | \mathbf{r})$  is called maximum a-posteriori probability (MAP) decoder.
- In 1974, Bahl, Cocke, Jelinek, and Raviv introduced a MAP decoder that can be applied to any linear code. This is known as BCJR algorithm.
- The BCJR algorithm computes the a-posteriori L-values (APP L-value) of each information bit.

$$L(u_i) = \ln \left[ \frac{P(u_i = +1 | \mathbf{r})}{P(u_i = -1 | \mathbf{r})} \right] \quad (1)$$

Now the complicity of this algorithm was much higher than Viterbi algorithm and that is why it was not popular in 70's, but in late 90's with, when this concatenated codes, turbo codes came into picture and we required soft estimates then these algorithms became very, very popular.

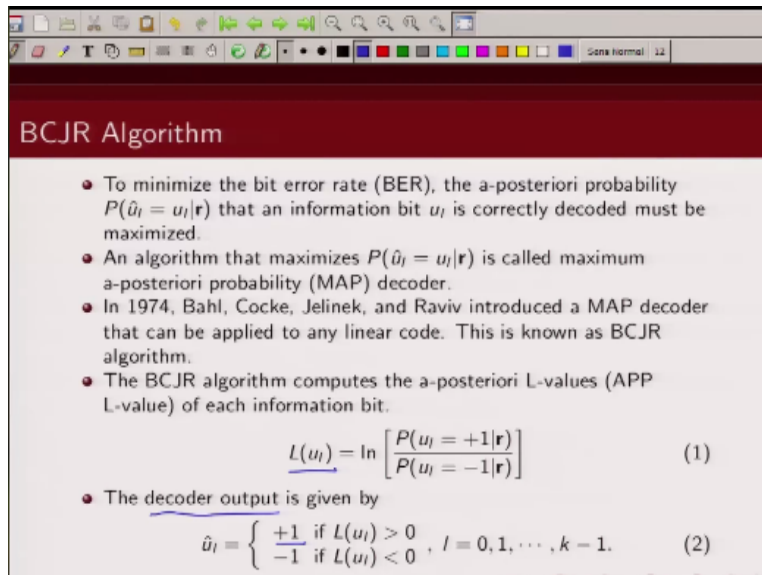
(Refer Slide Time: 02:38)



The image shows a presentation slide with a dark red header containing the text "BCJR Algorithm". Below the header, there is a list of four bullet points. The first bullet point states that to minimize the bit error rate (BER), the a-posteriori probability  $P(\hat{u}_l = u_l | \mathbf{r})$  must be maximized. The second bullet point identifies an algorithm that maximizes this probability as the maximum a-posteriori probability (MAP) decoder. The third bullet point mentions that in 1974, Bahl, Cocke, Jelinek, and Raviv introduced a MAP decoder for any linear code, known as the BCJR algorithm. The fourth bullet point states that the BCJR algorithm computes the a-posteriori L-values (APP L-value) for each information bit. Below the list, the equation 
$$L(u_l) = \ln \left[ \frac{P(u_l = +1 | \mathbf{r})}{P(u_l = -1 | \mathbf{r})} \right] \quad (1)$$
 is displayed. The slide also features a standard software toolbar at the top.

So what this algorithm does, it computes the a-posteriori probability, so I define a a-posteriori log likelihood value, I call it L-value like this. So it basically computes probability of  $u_l$  being +1 given a received sequence  $\mathbf{r}/P(u_l)$  being -1 given received sequence  $\mathbf{r}$ , take a log of that. Now if this L-value is greater than zero, then you decide in favor of  $u_l$  being +1, otherwise you decide in favor of  $u_l$  being -1.

(Refer Slide Time: 03:14)



The image shows a presentation slide with a dark red header containing the text "BCJR Algorithm". Below the header, there is a list of four bullet points. The first bullet point states that to minimize the bit error rate (BER), the a-posteriori probability  $P(\hat{u}_l = u_l | \mathbf{r})$  must be maximized. The second bullet point defines an algorithm that maximizes this probability as a maximum a-posteriori probability (MAP) decoder. The third bullet point mentions that in 1974, Bahl, Cocke, Jelinek, and Raviv introduced a MAP decoder for linear codes, known as the BCJR algorithm. The fourth bullet point states that the BCJR algorithm computes a-posteriori L-values (APP L-values) for each information bit. Below the list, equation (1) defines the L-value as the natural logarithm of the ratio of the a-posteriori probabilities for  $u_l = +1$  and  $u_l = -1$ . The final bullet point states that the decoder output is given by equation (2), which is a piecewise function that outputs +1 if the L-value is greater than 0 and -1 if it is less than 0, for  $l = 0, 1, \dots, k-1$ .

**BCJR Algorithm**

- To minimize the bit error rate (BER), the a-posteriori probability  $P(\hat{u}_l = u_l | \mathbf{r})$  that an information bit  $u_l$  is correctly decoded must be maximized.
- An algorithm that maximizes  $P(\hat{u}_l = u_l | \mathbf{r})$  is called maximum a-posteriori probability (MAP) decoder.
- In 1974, Bahl, Cocke, Jelinek, and Raviv introduced a MAP decoder that can be applied to any linear code. This is known as BCJR algorithm.
- The BCJR algorithm computes the a-posteriori L-values (APP L-value) of each information bit.

$$L(u_l) = \ln \left[ \frac{P(u_l = +1 | \mathbf{r})}{P(u_l = -1 | \mathbf{r})} \right] \quad (1)$$

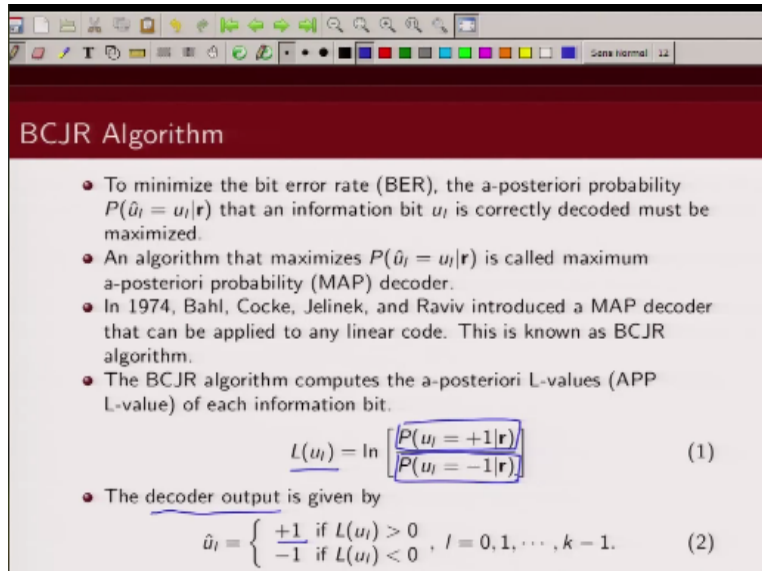
- The decoder output is given by

$$\hat{u}_l = \begin{cases} +1 & \text{if } L(u_l) > 0 \\ -1 & \text{if } L(u_l) < 0 \end{cases}, \quad l = 0, 1, \dots, k-1. \quad (2)$$

So your decoder output will be +1 if the L-value is greater than zero, otherwise you decide in favor of -1.



(Refer Slide Time: 03:32)



The image shows a presentation slide with a dark red header containing the text "BCJR Algorithm". Below the header, there is a list of four bullet points. The third bullet point is followed by a mathematical equation (1) for the L-value. The fourth bullet point is followed by a piecewise function (2) for the decoder output. The slide is displayed in a window with a standard toolbar at the top.

### BCJR Algorithm

- To minimize the bit error rate (BER), the a-posteriori probability  $P(\hat{u}_l = u_l | \mathbf{r})$  that an information bit  $u_l$  is correctly decoded must be maximized.
- An algorithm that maximizes  $P(\hat{u}_l = u_l | \mathbf{r})$  is called maximum a-posteriori probability (MAP) decoder.
- In 1974, Bahl, Cocke, Jelinek, and Raviv introduced a MAP decoder that can be applied to any linear code. This is known as BCJR algorithm.
- The BCJR algorithm computes the a-posteriori L-values (APP L-value) of each information bit.

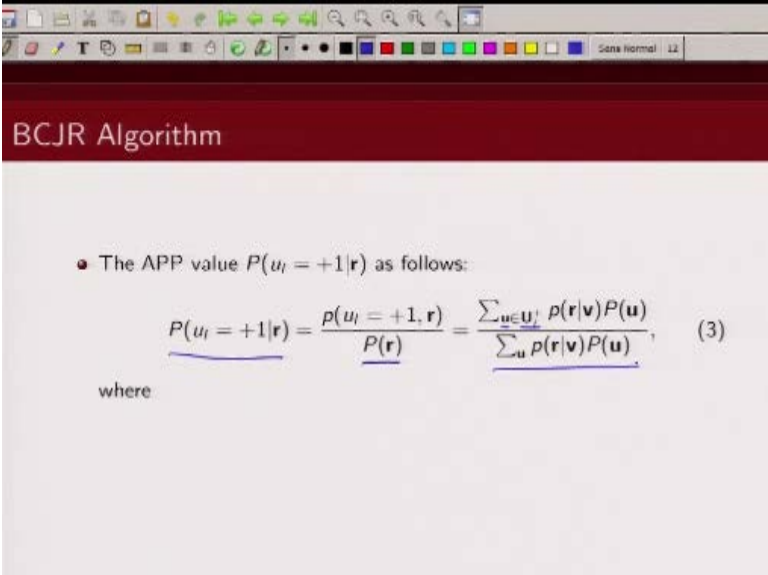
$$L(u_l) = \ln \left[ \frac{P(u_l = +1 | \mathbf{r})}{P(u_l = -1 | \mathbf{r})} \right] \quad (1)$$

- The decoder output is given by

$$\hat{u}_l = \begin{cases} +1 & \text{if } L(u_l) > 0 \\ -1 & \text{if } L(u_l) < 0 \end{cases}, \quad l = 0, 1, \dots, k-1. \quad (2)$$

So we are going to now talk about how to compute these terms. These terms you see in computation of APP value, how do we compute these terms and how we can exploit this structure of the trellis of the convolutional encoder to simplify this expression.

(Refer Slide Time: 03:53)



The slide is titled "BCJR Algorithm" and contains the following text and equation:

- The APP value  $P(u_i = +1 | \mathbf{r})$  as follows:

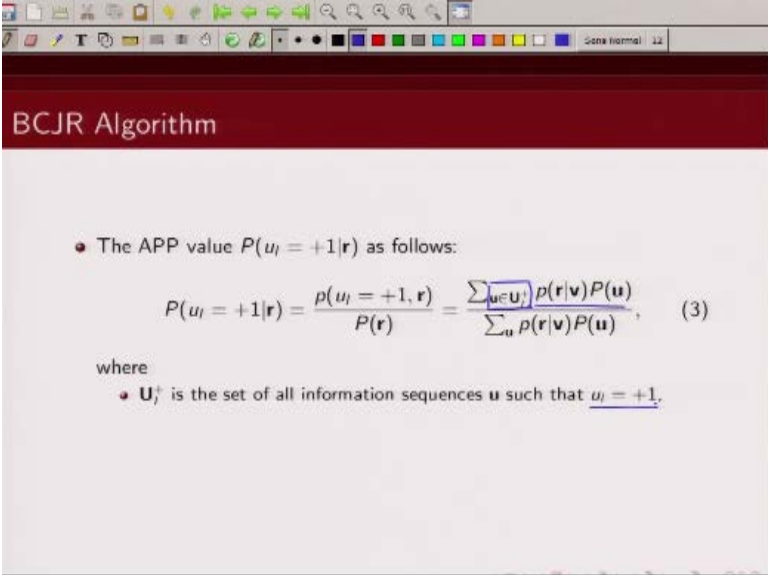
$$P(u_i = +1 | \mathbf{r}) = \frac{p(u_i = +1, \mathbf{r})}{P(\mathbf{r})} = \frac{\sum_{\mathbf{u} \in \mathbf{U}_i^+} p(\mathbf{r} | \mathbf{v}) P(\mathbf{u})}{\sum_{\mathbf{u}} p(\mathbf{r} | \mathbf{v}) P(\mathbf{u})}, \quad (3)$$

where

So let us look at this probability of  $u_1$  being +1 given a received sequence  $\mathbf{r}$ , this can be written as joint probability of  $u_1$  being +1 and received sequence  $\mathbf{r}$  divided by the probability of receiving this  $\mathbf{r}$ . Now this probability of  $u_1$  being plus one given a received sequence  $\mathbf{r}$  can be written as probability of  $\mathbf{r}$  given  $\mathbf{v}$ , multiplied probability of  $\mathbf{u}$  some over all input sequences that belongs to the set where  $u_1$  is +1.

And this can be written as probability of  $\mathbf{r}$  given  $\mathbf{v}$  multiplied by probability of  $\mathbf{u}$  some over all input sequences.

(Refer Slide Time: 04:36)



The slide is titled "BCJR Algorithm" and contains the following content:

- The APP value  $P(u_i = +1|\mathbf{r})$  as follows:

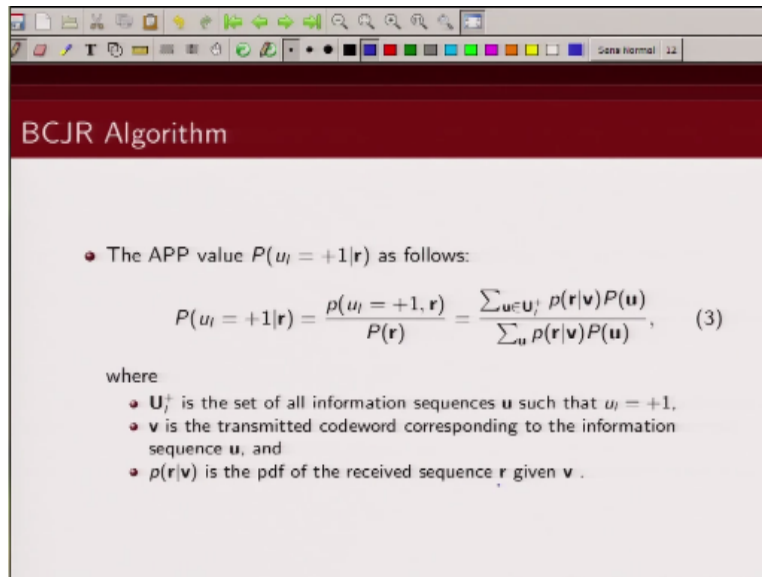
$$P(u_i = +1|\mathbf{r}) = \frac{p(u_i = +1, \mathbf{r})}{P(\mathbf{r})} = \frac{\sum_{\mathbf{u} \in \mathbf{U}_i^+} p(\mathbf{r}|\mathbf{v})P(\mathbf{u})}{\sum_{\mathbf{u}} p(\mathbf{r}|\mathbf{v})P(\mathbf{u})}, \quad (3)$$

where

- $\mathbf{U}_i^+$  is the set of all information sequences  $\mathbf{u}$  such that  $u_i = +1$ .

So as I said since we are interested in joint probability of  $u_i$  being +1 and  $\mathbf{r}$ , we sum this probability over all those set of information sequences where the bit, the corresponding bit is +1.

(Refer Slide Time: 05:40)



BCJR Algorithm

- The APP value  $P(u_i = +1|\mathbf{r})$  as follows:

$$P(u_i = +1|\mathbf{r}) = \frac{p(u_i = +1, \mathbf{r})}{P(\mathbf{r})} = \frac{\sum_{\mathbf{u} \in \mathbf{U}_i^+} p(\mathbf{r}|\mathbf{v})P(\mathbf{u})}{\sum_{\mathbf{u}} p(\mathbf{r}|\mathbf{v})P(\mathbf{u})}, \quad (3)$$

where

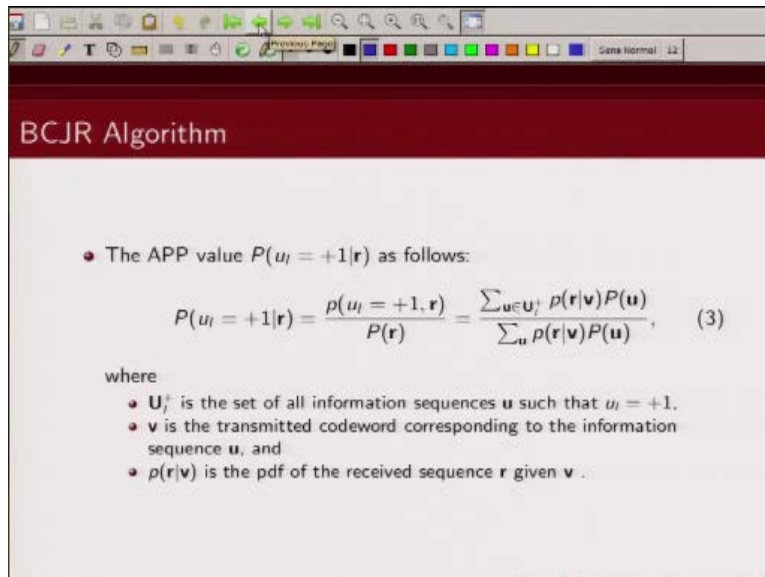
- $\mathbf{U}_i^+$  is the set of all information sequences  $\mathbf{u}$  such that  $u_i = +1$ ,
- $\mathbf{v}$  is the transmitted codeword corresponding to the information sequence  $\mathbf{u}$ , and
- $p(\mathbf{r}|\mathbf{v})$  is the pdf of the received sequence  $\mathbf{r}$  given  $\mathbf{v}$ .

And our transmitted code word is  $\mathbf{v}$ , our information sequence is  $\mathbf{u}$  and  $\mathbf{r}$  is the received sequence. So probability of  $\mathbf{r}$  given  $\mathbf{v}$  can be computed from the channel, given channel.

(Refer Slide Time: 05:22)

Similarly we can also compute.

(Refer Slide Time: 05:25)



BCJR Algorithm

- The APP value  $P(u_i = +1|\mathbf{r})$  as follows:

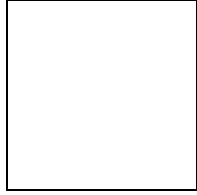
$$P(u_i = +1|\mathbf{r}) = \frac{p(u_i = +1, \mathbf{r})}{P(\mathbf{r})} = \frac{\sum_{\mathbf{u} \in \mathbf{U}_i^+} p(\mathbf{r}|\mathbf{v})P(\mathbf{u})}{\sum_{\mathbf{u}} p(\mathbf{r}|\mathbf{v})P(\mathbf{u})}, \quad (3)$$

where

- $\mathbf{U}_i^+$  is the set of all information sequences  $\mathbf{u}$  such that  $u_i = +1$ .
- $\mathbf{v}$  is the transmitted codeword corresponding to the information sequence  $\mathbf{u}$ , and
- $p(\mathbf{r}|\mathbf{v})$  is the pdf of the received sequence  $\mathbf{r}$  given  $\mathbf{v}$ .

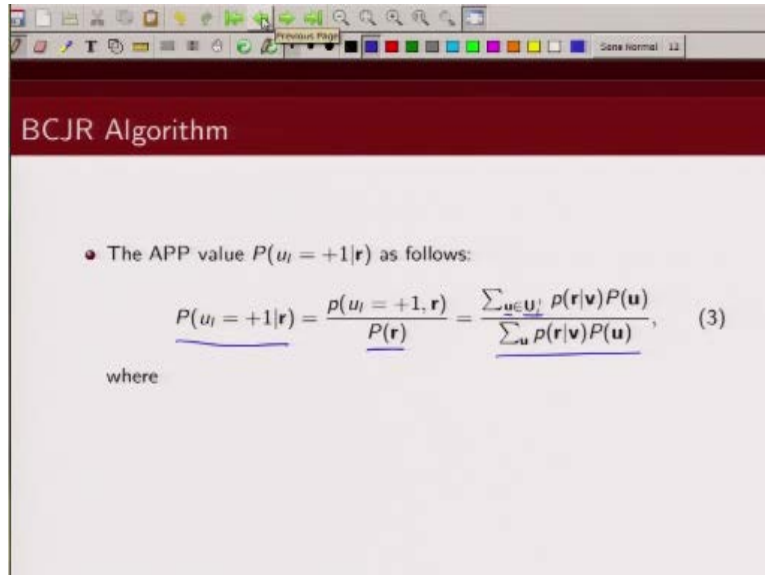
(Refer Slide Time: 05:26)

(Refer Slide Time: 05:26)



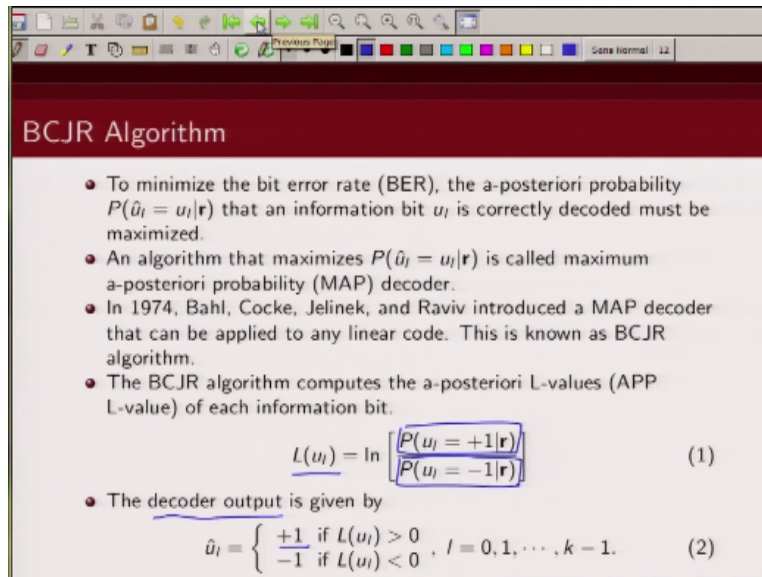


(Refer Slide Time: 05:27)



The image shows a screenshot of a presentation slide. At the top, there is a dark red header with the text "BCJR Algorithm" in white. Below the header, there is a white area containing a bullet point and a mathematical equation. The bullet point states: "• The APP value  $P(u_i = +1|\mathbf{r})$  as follows:". Below this, the equation is written as: 
$$\underline{P(u_i = +1|\mathbf{r})} = \frac{\underline{p(u_i = +1, \mathbf{r})}}{\underline{P(\mathbf{r})}} = \frac{\sum_{\mathbf{u} \in \mathcal{U}_i^+} \rho(\mathbf{r}|\mathbf{v})P(\mathbf{u})}{\sum_{\mathbf{u}} \rho(\mathbf{r}|\mathbf{v})P(\mathbf{u})}, \quad (3)$$
 Below the equation, the word "where" is written.

(Refer Slide Time: 05:28)



The image shows a presentation slide with a dark red header containing the title "BCJR Algorithm". Below the header, there is a list of four bullet points. The third bullet point is followed by a mathematical equation (1) for the L-value. The fourth bullet point is followed by a piecewise function (2) for the decoder output. The slide is displayed in a window with a standard toolbar at the top.

### BCJR Algorithm

- To minimize the bit error rate (BER), the a-posteriori probability  $P(\hat{u}_l = u_l | \mathbf{r})$  that an information bit  $u_l$  is correctly decoded must be maximized.
- An algorithm that maximizes  $P(\hat{u}_l = u_l | \mathbf{r})$  is called maximum a-posteriori probability (MAP) decoder.
- In 1974, Bahl, Cocke, Jelinek, and Raviv introduced a MAP decoder that can be applied to any linear code. This is known as BCJR algorithm.
- The BCJR algorithm computes the a-posteriori L-values (APP L-value) of each information bit.

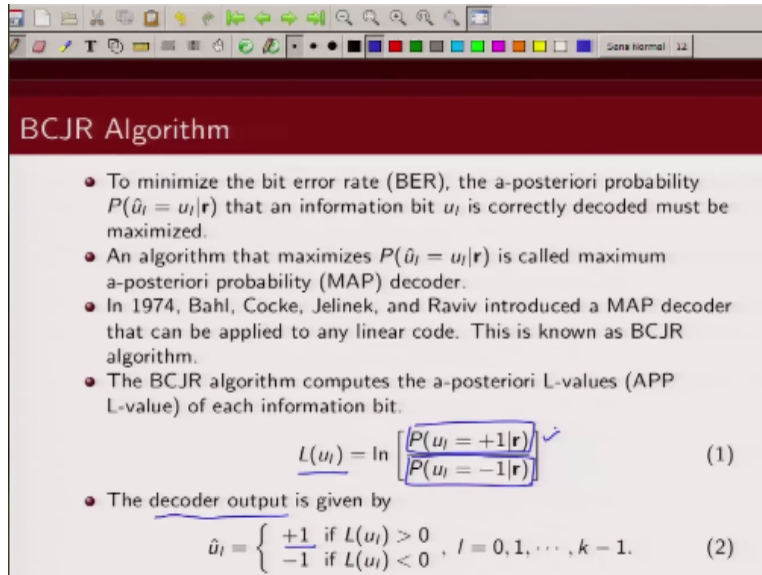
$$L(u_l) = \ln \frac{P(u_l = +1 | \mathbf{r})}{P(u_l = -1 | \mathbf{r})} \quad (1)$$

- The decoder output is given by

$$\hat{u}_l = \begin{cases} +1 & \text{if } L(u_l) > 0 \\ -1 & \text{if } L(u_l) < 0 \end{cases}, l = 0, 1, \dots, k-1. \quad (2)$$

Now if you go back here the denominator we need to compute probability of  $u_l$  being -1 given  $\mathbf{r}$ .

(Refer Slide Time: 05:36)



The image shows a presentation slide with a dark red header containing the text "BCJR Algorithm". Below the header, there is a list of four bullet points. The third bullet point is followed by a mathematical equation (1) for the a-posteriori L-value. The fourth bullet point is followed by a piecewise function (2) for the decoder output. The slide is displayed in a window with a standard toolbar at the top.

### BCJR Algorithm

- To minimize the bit error rate (BER), the a-posteriori probability  $P(\hat{u}_l = u_l | \mathbf{r})$  that an information bit  $u_l$  is correctly decoded must be maximized.
- An algorithm that maximizes  $P(\hat{u}_l = u_l | \mathbf{r})$  is called maximum a-posteriori probability (MAP) decoder.
- In 1974, Bahl, Cocke, Jelinek, and Raviv introduced a MAP decoder that can be applied to any linear code. This is known as BCJR algorithm.
- The BCJR algorithm computes the a-posteriori L-values (APP L-value) of each information bit.

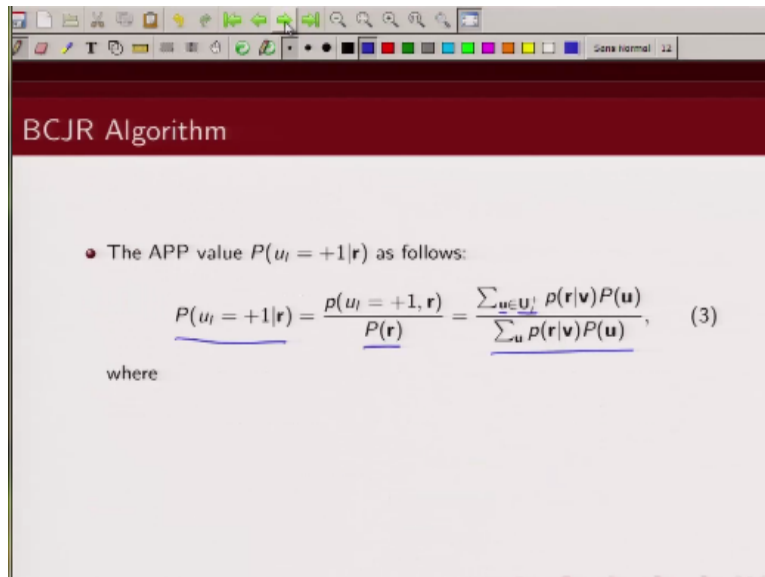
$$L(u_l) = \ln \left[ \frac{P(u_l = +1 | \mathbf{r})}{P(u_l = -1 | \mathbf{r})} \right] \quad (1)$$

- The decoder output is given by

$$\hat{u}_l = \begin{cases} +1 & \text{if } L(u_l) > 0 \\ -1 & \text{if } L(u_l) < 0 \end{cases}, \quad l = 0, 1, \dots, k-1. \quad (2)$$

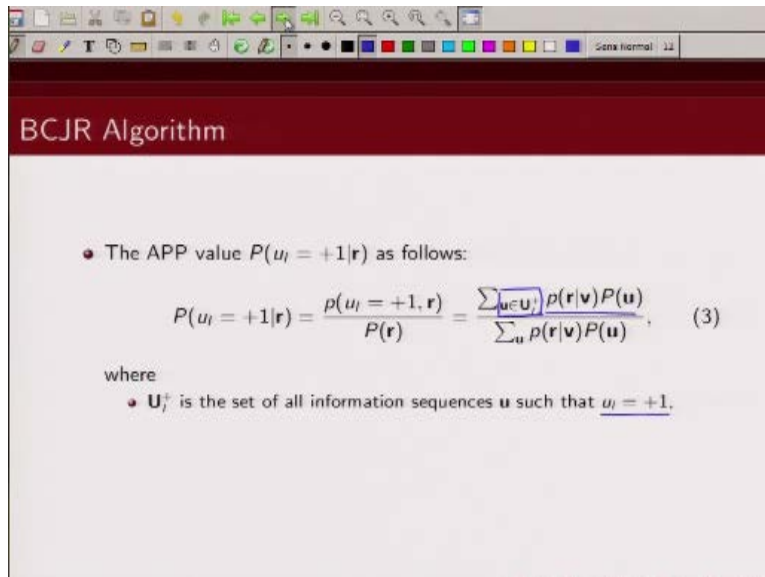
So similar to this term.

(Refer Slide Time: 05:37)



The image shows a presentation slide with a dark red header containing the text "BCJR Algorithm". Below the header, there is a bullet point: "• The APP value  $P(u_t = +1 | \mathbf{r})$  as follows:". Below the bullet point is a mathematical equation labeled (3):
$$P(u_t = +1 | \mathbf{r}) = \frac{p(u_t = +1, \mathbf{r})}{P(\mathbf{r})} = \frac{\sum_{\mathbf{u} \in \mathbf{U}_t} p(\mathbf{r} | \mathbf{v}) P(\mathbf{u})}{\sum_{\mathbf{u}} p(\mathbf{r} | \mathbf{v}) P(\mathbf{u})}, \quad (3)$$
The terms  $P(u_t = +1 | \mathbf{r})$ ,  $P(\mathbf{r})$ , and the denominator of the fraction are underlined in blue. Below the equation, the word "where" is written.

(Refer Slide Time: 05:39)



BCJR Algorithm

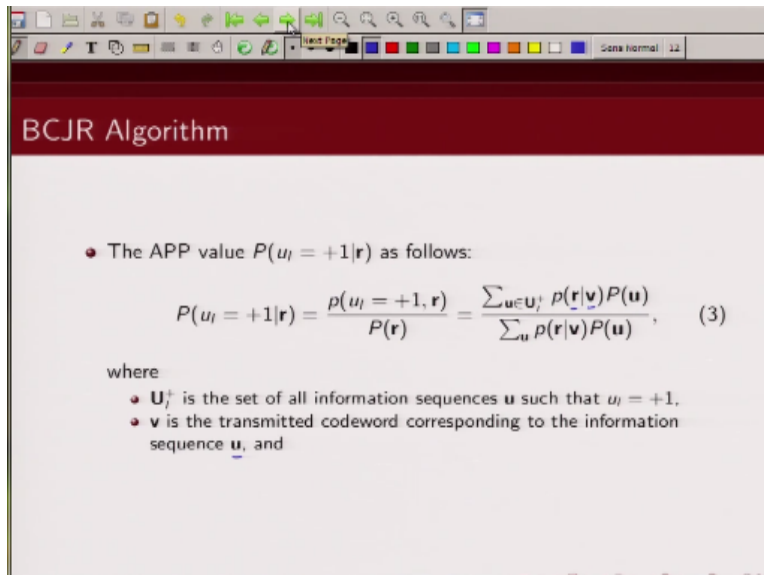
- The APP value  $P(u_i = +1|\mathbf{r})$  as follows:

$$P(u_i = +1|\mathbf{r}) = \frac{p(u_i = +1, \mathbf{r})}{P(\mathbf{r})} = \frac{\sum_{\mathbf{u} \in \mathbf{U}_i^+} p(\mathbf{r}|\mathbf{v})P(\mathbf{u})}{\sum_{\mathbf{u}} p(\mathbf{r}|\mathbf{v})P(\mathbf{u})}, \quad (3)$$

where

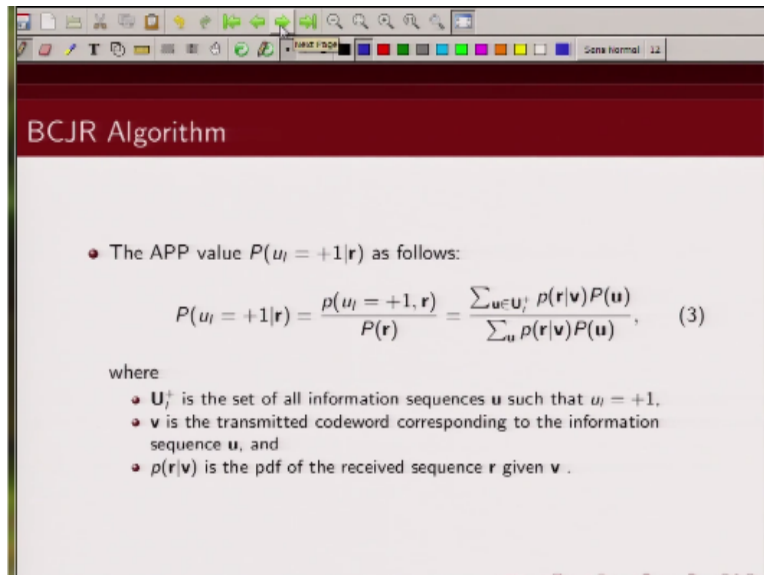
- $\mathbf{U}_i^+$  is the set of all information sequences  $\mathbf{u}$  such that  $u_i = +1$ .

(Refer Slide Time: 05:41)



The image shows a presentation slide with a dark red header containing the text "BCJR Algorithm". Below the header, there is a list item: "• The APP value  $P(u_i = +1|\mathbf{r})$  as follows:". This is followed by a mathematical equation: 
$$P(u_i = +1|\mathbf{r}) = \frac{p(u_i = +1, \mathbf{r})}{P(\mathbf{r})} = \frac{\sum_{\mathbf{u} \in \mathbf{U}_i^+} p(\mathbf{r}|\mathbf{v})P(\mathbf{u})}{\sum_{\mathbf{u}} p(\mathbf{r}|\mathbf{v})P(\mathbf{u})}, \quad (3)$$
 Below the equation, the word "where" is used to introduce two bullet points: "•  $\mathbf{U}_i^+$  is the set of all information sequences  $\mathbf{u}$  such that  $u_i = +1$ ." and "•  $\mathbf{v}$  is the transmitted codeword corresponding to the information sequence  $\mathbf{u}$ , and".

(Refer Slide Time: 05:42)



**BCJR Algorithm**

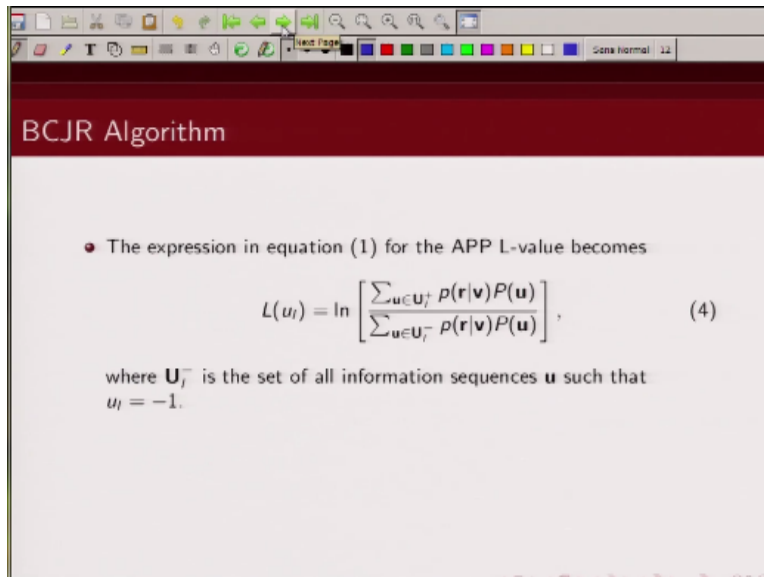
- The APP value  $P(u_i = +1|\mathbf{r})$  as follows:

$$P(u_i = +1|\mathbf{r}) = \frac{p(u_i = +1, \mathbf{r})}{P(\mathbf{r})} = \frac{\sum_{\mathbf{u} \in \mathbf{U}_i^+} p(\mathbf{r}|\mathbf{v})P(\mathbf{u})}{\sum_{\mathbf{u}} p(\mathbf{r}|\mathbf{v})P(\mathbf{u})}, \quad (3)$$

where

- $\mathbf{U}_i^+$  is the set of all information sequences  $\mathbf{u}$  such that  $u_i = +1$ .
- $\mathbf{v}$  is the transmitted codeword corresponding to the information sequence  $\mathbf{u}$ , and
- $p(\mathbf{r}|\mathbf{v})$  is the pdf of the received sequence  $\mathbf{r}$  given  $\mathbf{v}$ .

(Refer Slide Time: 05:43)

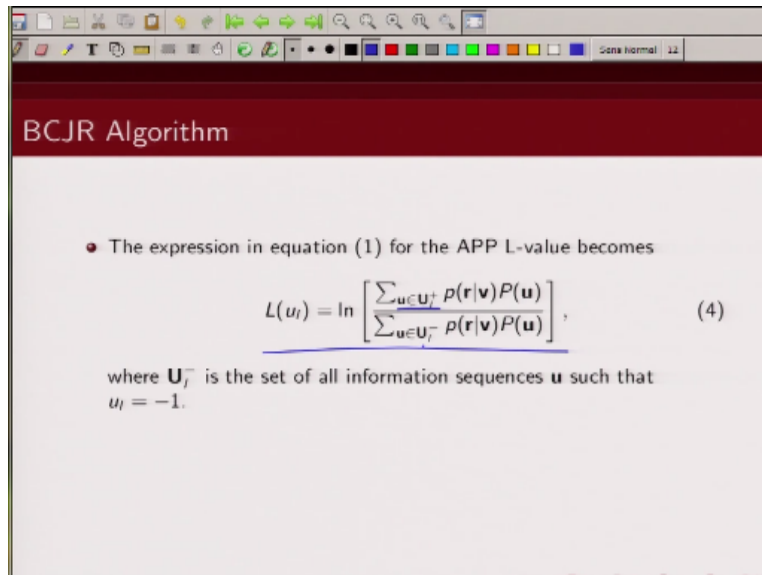


The image shows a screenshot of a presentation slide. At the top, there is a dark red header with the text "BCJR Algorithm" in white. Below the header, there is a bullet point that reads: "• The expression in equation (1) for the APP L-value becomes". Underneath the bullet point is a mathematical equation labeled (4):
$$L(u_i) = \ln \left[ \frac{\sum_{\mathbf{u} \in \mathbf{U}_i^+} p(\mathbf{r}|\mathbf{v})P(\mathbf{u})}{\sum_{\mathbf{u} \in \mathbf{U}_i^-} p(\mathbf{r}|\mathbf{v})P(\mathbf{u})} \right], \quad (4)$$
Below the equation, there is a text explanation: "where  $\mathbf{U}_i^-$  is the set of all information sequences  $\mathbf{u}$  such that  $u_i = -1$ ." The slide is displayed in a window with a standard toolbar at the top.

We can also write probability of  $u_i$  being -1 given  $r$  and probability of  $r$  is a common term.



(Refer Slide Time: 05:48)

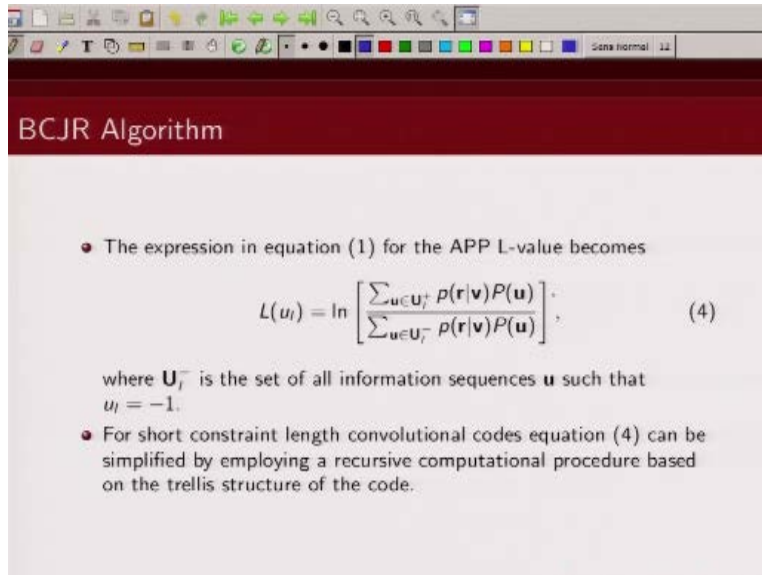


The screenshot shows a presentation slide with a dark red header containing the text "BCJR Algorithm". Below the header, there is a bullet point that reads: "The expression in equation (1) for the APP L-value becomes". This is followed by the equation (4): 
$$L(u_i) = \ln \left[ \frac{\sum_{\mathbf{u} \in \mathbf{U}_i^+} p(\mathbf{r}|\mathbf{v})P(\mathbf{u})}{\sum_{\mathbf{u} \in \mathbf{U}_i^-} p(\mathbf{r}|\mathbf{v})P(\mathbf{u})} \right], \quad (4)$$
 The equation is underlined. Below the equation, the text states: "where  $\mathbf{U}_i^-$  is the set of all information sequences  $\mathbf{u}$  such that  $u_i = -1$ ."

So if we do that what we get is this. So again this L-value, the APP value of  $u_i$  is given by probability of  $r$  given  $v$  multiply the probability of  $u$  where we are summing over all information sequences where the corresponding bit is +1. And similarly for the denominator we are summing over all information sequences where information bit is -1.

We will illustrate this with help of an example and then things will be little more clear.

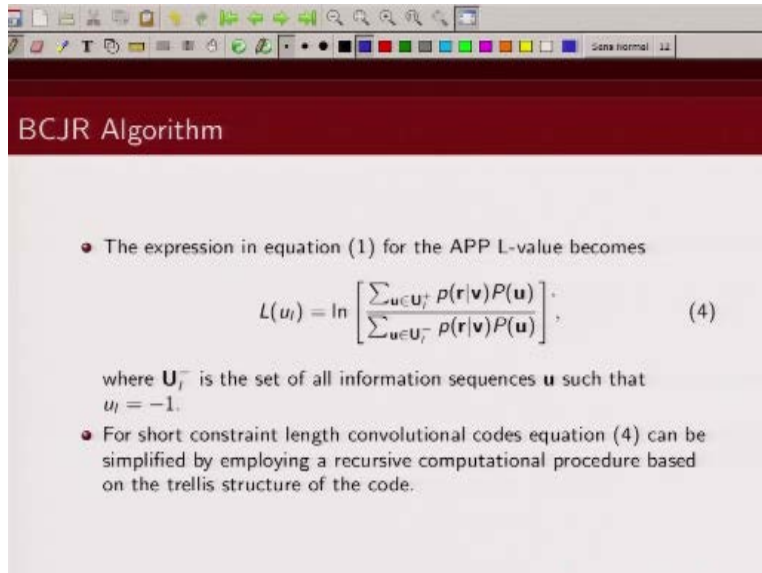
(Refer Slide Time: 06:34)



The image shows a presentation slide with a dark red header containing the text "BCJR Algorithm". Below the header, there is a bulleted list item: "• The expression in equation (1) for the APP L-value becomes". This is followed by a mathematical equation: 
$$L(u_i) = \ln \left[ \frac{\sum_{\mathbf{u} \in \mathbf{U}_i^+} \rho(\mathbf{r}|\mathbf{v})P(\mathbf{u})}{\sum_{\mathbf{u} \in \mathbf{U}_i^-} \rho(\mathbf{r}|\mathbf{v})P(\mathbf{u})} \right], \quad (4)$$
 Below the equation, there is a text explanation: "where  $\mathbf{U}_i^-$  is the set of all information sequences  $\mathbf{u}$  such that  $u_i = -1$ ." This is followed by another bulleted list item: "• For short constraint length convolutional codes equation (4) can be simplified by employing a recursive computational procedure based on the trellis structure of the code." The slide is displayed in a window with a standard operating system toolbar at the top.

Now note here, if you have very large sequences this is some over all input sequences where  $u_i$  is +1 in this somewhat all input sequences where  $u_i$  is -1. So if your information sequence is large, this is somewhat very large number of possibilities. So this is quite complex, now can we use the structure of the convolutional code to simplify this expression.

(Refer Slide Time: 07:02)



The image shows a presentation slide with a dark red header containing the text "BCJR Algorithm". Below the header, there is a bulleted list. The first bullet point states: "The expression in equation (1) for the APP L-value becomes". This is followed by equation (4): 
$$L(u_i) = \ln \left[ \frac{\sum_{\mathbf{u} \in \mathbf{U}_i^+} \rho(\mathbf{r}|\mathbf{v})P(\mathbf{u})}{\sum_{\mathbf{u} \in \mathbf{U}_i^-} \rho(\mathbf{r}|\mathbf{v})P(\mathbf{u})} \right], \quad (4)$$
 Below the equation, it says "where  $\mathbf{U}_i^-$  is the set of all information sequences  $\mathbf{u}$  such that  $u_i = -1$ ." The second bullet point states: "For short constraint length convolutional codes equation (4) can be simplified by employing a recursive computational procedure based on the trellis structure of the code." The slide is displayed in a window with a standard operating system toolbar at the top.

- The expression in equation (1) for the APP L-value becomes

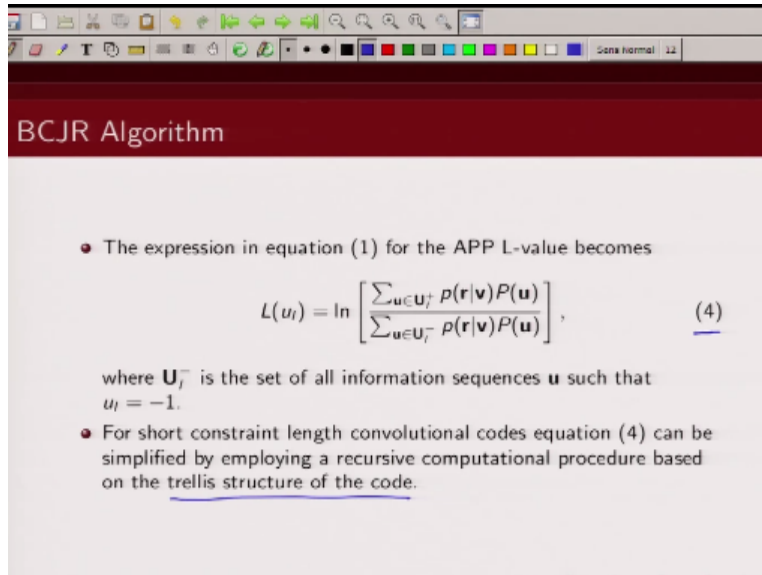
$$L(u_i) = \ln \left[ \frac{\sum_{\mathbf{u} \in \mathbf{U}_i^+} \rho(\mathbf{r}|\mathbf{v})P(\mathbf{u})}{\sum_{\mathbf{u} \in \mathbf{U}_i^-} \rho(\mathbf{r}|\mathbf{v})P(\mathbf{u})} \right], \quad (4)$$

where  $\mathbf{U}_i^-$  is the set of all information sequences  $\mathbf{u}$  such that  $u_i = -1$ .

- For short constraint length convolutional codes equation (4) can be simplified by employing a recursive computational procedure based on the trellis structure of the code.

The answer to this is yes, so we are going to basically simplify this equation for by using the trellis structure of the convolutional code. We know all possible transitions are not possible. So our trellis diagram what the state diagram will ensure, will tell us what are the valid transitions.

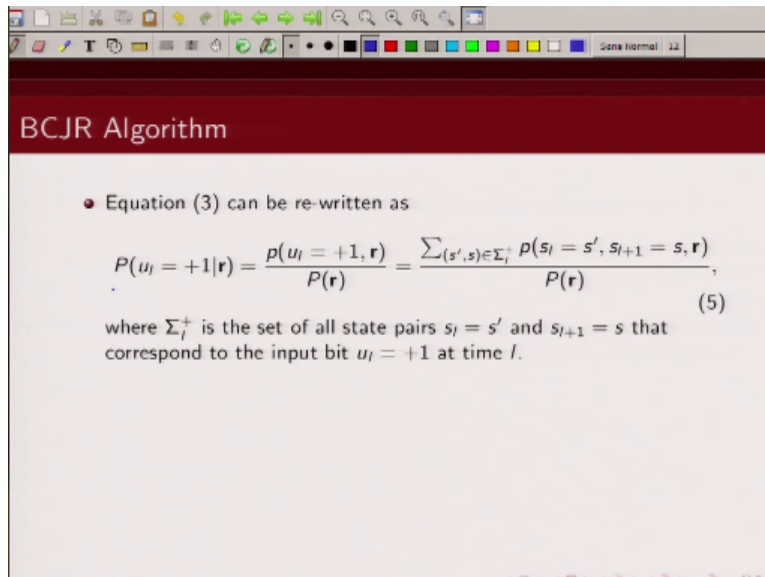
(Refer Slide Time: 07:29)



The image shows a presentation slide with a dark red header containing the text "BCJR Algorithm". Below the header, there is a list of bullet points. The first bullet point states: "The expression in equation (1) for the APP L-value becomes". This is followed by equation (4): 
$$L(u_i) = \ln \left[ \frac{\sum_{\mathbf{u} \in \mathbf{U}_i^+} p(\mathbf{r}|\mathbf{v})P(\mathbf{u})}{\sum_{\mathbf{u} \in \mathbf{U}_i^-} p(\mathbf{r}|\mathbf{v})P(\mathbf{u})} \right], \quad (4)$$
 where  $\mathbf{U}_i^-$  is the set of all information sequences  $\mathbf{u}$  such that  $u_i = -1$ . The second bullet point states: "For short constraint length convolutional codes equation (4) can be simplified by employing a recursive computational procedure based on the trellis structure of the code." The slide also features a standard software toolbar at the top with various icons for navigation and editing.

So we can simplify this expression using our valid state transitions. So what we are going to do is we are going to make use of the trellis structure of the code to simplify our equation number 4.

(Refer Slide Time: 07:49)



The image shows a presentation slide with a dark red header containing the text "BCJR Algorithm". Below the header, there is a bullet point stating "Equation (3) can be re-written as". This is followed by a mathematical equation labeled (5):

$$P(u_l = +1|\mathbf{r}) = \frac{p(u_l = +1, \mathbf{r})}{P(\mathbf{r})} = \frac{\sum_{(s', s) \in \Sigma_l^+} p(s_l = s', s_{l+1} = s, \mathbf{r})}{P(\mathbf{r})}, \quad (5)$$

where  $\Sigma_l^+$  is the set of all state pairs  $s_l = s'$  and  $s_{l+1} = s$  that correspond to the input bit  $u_l = +1$  at time  $l$ .

So let us see how do we do it.

(Refer Slide Time: 07:54)

**BCJR Algorithm**

- Equation (3) can be re-written as

$$P(u_l = +1 | \mathbf{r}) = \frac{P(u_l = +1, \mathbf{r})}{P(\mathbf{r})} = \frac{\sum_{(s', s) \in \Sigma_l^+} P(s_l = s', s_{l+1} = s, \mathbf{r})}{P(\mathbf{r})}, \quad (5)$$

where  $\Sigma_l^+$  is the set of all state pairs  $s_l = s'$  and  $s_{l+1} = s$  that correspond to the input bit  $u_l = +1$  at time  $l$ .

So again we go back and look at this probability of  $u_l$  being +1 given our received sequence  $\mathbf{r}$  as we have written, this can be written as joint probability of  $u_l$  being 1 and the probability of receiving  $\mathbf{r}$  divided by the probability of  $\mathbf{r}$ . Now we are going to now look at this expression, this is joint probability of  $u_l$  being +1 and given the received sequence  $\mathbf{r}$  has been received.

So if you look at any trellis diagram let us say this is on trellis diagram, this simple two state code like that you have. So we are interested in where  $u_l$  is +1 and where  $u_l$  is -1. Let us say this is 0/00, this is 1/11, this is – let us say 1/10, this is 0/01. So let us look at one trellis section, so we are interested in all those transitions which belongs to  $u_l+1$ . Now what are those transitions, so in this example this is one set transition and the other is this transition okay.

So what I am writing here is then I am interested in what is the joint probability that the previous state is  $S'$  s prime, the next state is  $S$ , and the received sequence is  $\mathbf{r}$  and I am summing over all those state transitions that belong to the set pair where the input corresponds to this transition is +1. So note what is my this  $\Sigma_l^+$  it is the set of all state pairs where the initial state is  $S'$  then next state is  $s$ , so it is a pair of states where the transitions, the input bit corresponding to a valid transition is +1.

So in this case the set that belongs to this is given by this red line okay. So I can write the joint probability of  $u_l$  being +1 and  $\mathbf{r}$  in terms of condition on the valid trellis transitions in this way, I can write it as what is the probability that the initial state is  $s'$  next state is  $s$  given that I have received sequence  $\mathbf{r}$ . And I sum over all those transitions which belong to input bit being +1.

(Refer Slide Time: 11:09)

**BCJR Algorithm**

- Equation (3) can be re-written as
 
$$P(u_l = +1 | \mathbf{r}) = \frac{p(u_l = +1, \mathbf{r})}{P(\mathbf{r})} = \frac{\sum_{(s', s) \in \Sigma_l^+} p(s_l = s', s_{l+1} = s, \mathbf{r})}{P(\mathbf{r})}, \quad (5)$$
 where  $\Sigma_l^+$  is the set of all state pairs  $s_l = s'$  and  $s_{l+1} = s$  that correspond to the input bit  $u_l = +1$  at time  $l$ .
- Similarly, equation (4) can be written as
 
$$L(u_l) = \ln \left\{ \frac{\sum_{(s', s) \in \Sigma_l^+} p(s_l = s', s_{l+1} = s, \mathbf{r})}{\sum_{(s', s) \in \Sigma_l^-} p(s_l = s', s_{l+1} = s, \mathbf{r})} \right\}, \quad (6)$$
 where  $\Sigma_l^-$  is the set of all state pairs  $s_l = s'$  and  $s_{l+1} = s$  that correspond to the input bit  $u_l = -1$  at time  $l$ .

Similarly I can write exactly the way I wrote probability of  $u_l$  being +1 given  $\mathbf{r}$ , I can follow the same procedure to write what is the probability of  $u_l$  being -1 given  $\mathbf{r}$ . So what will be the change here, so I will compute this probability and I will sum over all those state pairs which corresponds to input bit -1.

(Refer Slide Time: 11:39)

**BCJR Algorithm**

- Equation (3) can be re-written as  $P(u_i = -1/r)$   $\sum_{(s',s) \in \Sigma_i^-}$

$$P(u_i = +1|r) = \frac{p(u_i = +1, \mathbf{r})}{P(\mathbf{r})} = \frac{\sum_{(s',s) \in \Sigma_i^+} p(s_i = s', s_{i+1} = s, \mathbf{r})}{P(\mathbf{r})}, \quad (5)$$

where  $\Sigma_i^+$  is the set of all state pairs  $s_i = s'$  and  $s_{i+1} = s$  that correspond to the input bit  $u_i = +1$  at time  $i$ .

- Similarly, equation (4) can be written as

$$L(u_i) = \ln \left\{ \frac{\sum_{(s',s) \in \Sigma_i^+} p(s_i = s', s_{i+1} = s, \mathbf{r})}{\sum_{(s',s) \in \Sigma_i^-} p(s_i = s', s_{i+1} = s, \mathbf{r})} \right\}, \quad (6)$$

where  $\Sigma_i^-$  is the set of all state pairs  $s_i = s'$  and  $s_{i+1} = s$  that correspond to the input bit  $u_i = -1$  at time  $i$ .

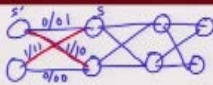
So if I plug these values of probabilities which are given by equation 5 and similarly I can calculate the probability of  $u_i$  being -1 given  $\mathbf{r}$ . So instead of this thing here, I will have  $\sum$  over  $(s', s)$   $\sum$  over all those pairs which corresponds to  $u_i$  being -1. And I will get this same thing here. So if I do that what I will get is equation number 6.



(Refer Slide Time: 12:15)

BCJR Algorithm

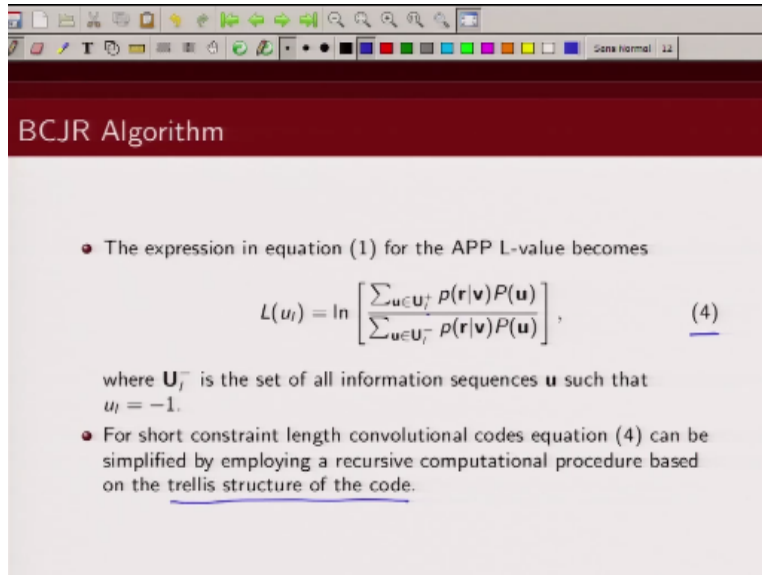
- Equation (3) can be re-written as


$$P(u_l = +1 | \mathbf{r}) = \frac{p(u_l = +1, \mathbf{r})}{P(\mathbf{r})} = \frac{\sum_{\{s', s\} \in \Sigma_l^+} p(s_l = s', s_{l+1} = s, \mathbf{r})}{P(\mathbf{r})}, \quad (5)$$

where  $\Sigma_l^+$  is the set of all state pairs  $s_l = s'$  and  $s_{l+1} = s$  that correspond to the input bit  $u_l = +1$  at time  $l$ .

So note that previously.

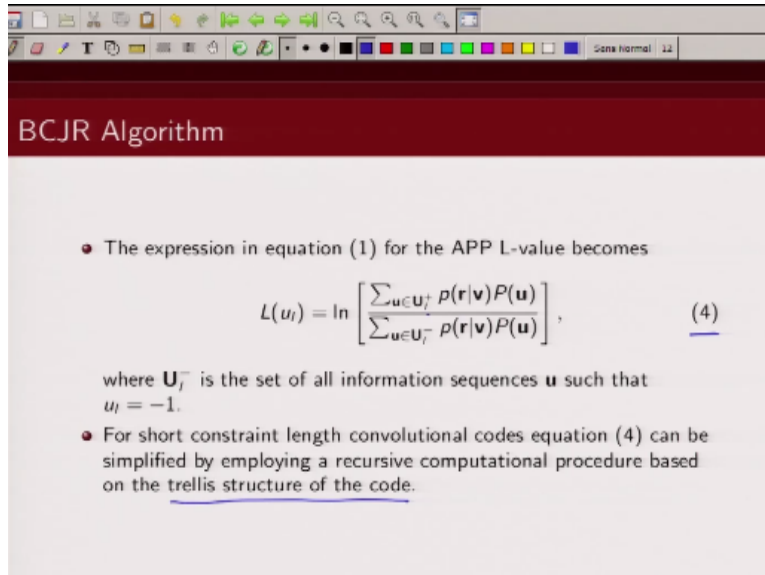
(Refer Slide Time: 12:16)



The image shows a presentation slide with a dark red header containing the text "BCJR Algorithm". Below the header, there is a bulleted list. The first bullet point states: "The expression in equation (1) for the APP L-value becomes". This is followed by equation (4): 
$$L(u_i) = \ln \left[ \frac{\sum_{\mathbf{u} \in \mathbf{U}_i^+} \rho(\mathbf{r}|\mathbf{v})P(\mathbf{u})}{\sum_{\mathbf{u} \in \mathbf{U}_i^-} \rho(\mathbf{r}|\mathbf{v})P(\mathbf{u})} \right], \quad (4)$$
 where  $\mathbf{U}_i^-$  is the set of all information sequences  $\mathbf{u}$  such that  $u_i = -1$ . The second bullet point states: "For short constraint length convolutional codes equation (4) can be simplified by employing a recursive computational procedure based on the trellis structure of the code." The slide also features a standard software toolbar at the top with various icons and a status bar at the bottom right showing "Sans Normal 12".

I had this same expression equation number four in terms of this input sequence  $u_i$ .

(Refer Slide Time: 12:29)



The image shows a presentation slide with a dark red header containing the text "BCJR Algorithm". Below the header, there is a list of bullet points. The first bullet point states: "The expression in equation (1) for the APP L-value becomes". This is followed by equation (4): 
$$L(u_i) = \ln \left[ \frac{\sum_{\mathbf{u} \in \mathbf{U}_i^+} \rho(\mathbf{r}|\mathbf{v})P(\mathbf{u})}{\sum_{\mathbf{u} \in \mathbf{U}_i^-} \rho(\mathbf{r}|\mathbf{v})P(\mathbf{u})} \right], \quad (4)$$
 where  $\mathbf{U}_i^-$  is the set of all information sequences  $\mathbf{u}$  such that  $u_i = -1$ . The second bullet point states: "For short constraint length convolutional codes equation (4) can be simplified by employing a recursive computational procedure based on the trellis structure of the code." The slide also features a standard software toolbar at the top with various icons for navigation and editing.

- The expression in equation (1) for the APP L-value becomes

$$L(u_i) = \ln \left[ \frac{\sum_{\mathbf{u} \in \mathbf{U}_i^+} \rho(\mathbf{r}|\mathbf{v})P(\mathbf{u})}{\sum_{\mathbf{u} \in \mathbf{U}_i^-} \rho(\mathbf{r}|\mathbf{v})P(\mathbf{u})} \right], \quad (4)$$

where  $\mathbf{U}_i^-$  is the set of all information sequences  $\mathbf{u}$  such that  $u_i = -1$ .

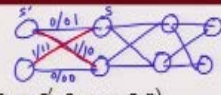
- For short constraint length convolutional codes equation (4) can be simplified by employing a recursive computational procedure based on the trellis structure of the code.

Now if our input sequence is very long this is  $\sum$  over, so large number of terms.

(Refer Slide Time: 12:31)

BCJR Algorithm

- Equation (3) can be re-written as


$$P(u_l = +1 | \mathbf{r}) = \frac{p(u_l = +1, \mathbf{r})}{P(\mathbf{r})} = \frac{\sum_{\{s', s\} \in \Sigma_l^+} p(s_l = s', s_{l+1} = s, \mathbf{r})}{P(\mathbf{r})}, \quad (5)$$

where  $\Sigma_l^+$  is the set of all state pairs  $s_l = s'$  and  $s_{l+1} = s$  that correspond to the input bit  $u_l = +1$  at time  $l$ .

Whereas I have now simplified my expression.

(Refer Slide Time: 12:37)

**BCJR Algorithm**

- Equation (3) can be re-written as  $P(u_l = -1/r)$   $\sum_{(s',s) \in \Sigma_l^-}$

$$P(u_l = +1|r) = \frac{p(u_l = +1, r)}{P(r)} = \frac{\sum_{(s',s) \in \Sigma_l^+} p(s_l = s', s_{l+1} = s, r)}{P(r)}, \quad (5)$$

where  $\Sigma_l^+$  is the set of all state pairs  $s_l = s'$  and  $s_{l+1} = s$  that correspond to the input bit  $u_l = +1$  at time  $l$ .

- Similarly, equation (4) can be written as

$$L(u_l) = \ln \left\{ \frac{\sum_{(s',s) \in \Sigma_l^+} p(s_l = s', s_{l+1} = s, r)}{\sum_{(s',s) \in \Sigma_l^-} p(s_l = s', s_{l+1} = s, r)} \right\}, \quad (6)$$

where  $\Sigma_l^-$  is the set of all state pairs  $s_l = s'$  and  $s_{l+1} = s$  that correspond to the input bit  $u_l = -1$  at time  $l$ .

So this  $\sum$  is now only over valid transitions corresponding to  $u_l$  being +1 and this  $\sum$  is over valid transitions corresponding to  $u_l$  being -1.

(Refer Slide Time: 12:52)

**BCJR Algorithm**

- Equation (3) can be re-written as  $P(u_i = -1/r)$   $\Sigma_{(s',s)} \Sigma_1^-$

$$P(u_i = +1|\mathbf{r}) = \frac{p(u_i = +1, \mathbf{r})}{P(\mathbf{r})} = \frac{\sum_{(s',s) \in \Sigma_i^+} p(s_i = s', s_{i+1} = s, \mathbf{r})}{P(\mathbf{r})}, \quad (5)$$

where  $\Sigma_i^+$  is the set of all state pairs  $s_i = s'$  and  $s_{i+1} = s$  that correspond to the input bit  $u_i = +1$  at time  $i$ .

- Similarly, equation (4) can be written as

$$L(u_i) = \ln \left\{ \frac{\sum_{(s',s) \in \Sigma_i^+} p(s_i = s', s_{i+1} = s, \mathbf{r})}{\sum_{(s',s) \in \Sigma_i^-} p(s_i = s', s_{i+1} = s, \mathbf{r})} \right\}, \quad (6)$$

where  $\Sigma_i^-$  is the set of all state pairs  $s_i = s'$  and  $s_{i+1} = s$  that correspond to the input bit  $u_i = -1$  at time  $i$ .

So I have simplified my equation number four, equation number six and I have used the state diagram or the trellis diagram of the convolutional encoder to simplify my expression. So this will be my a-posteriori probability log like L-value a-posteriori probability. Now how do I compute this term?

This we will show that if we can write this term as product of three terms and two of these terms can be computed recursively, that is what I am going to show in the subsequent slide. So let us look at this expression, how do we compute the probability that in the current state it is in  $s'$  next state is  $s$  given a received sequence  $\mathbf{r}$ .

(Refer Slide Time: 13:44)

### BCJR Algorithm

- The joint pdf's  $p(s', s, r)$  in equation (6) can be evaluated recursively

$$\begin{aligned}
 p(s', s, r) &= p(s', s, r_{t < l}, r_l, r_{t > l}) \\
 &= p(r_{t > l} | s', s, r_{t < l}, r_l) p(s', s, r_{t < l}, r_l) \\
 &= p(r_{t > l} | s', s, r_{t < l}, r_l) p(s, r_l | s', r_{t < l}) p(s', r_{t < l}) \\
 &= p(r_{t > l} | s) p(s, r_l | s') p(s', r_{t < l}), \quad (7)
 \end{aligned}$$

where  $r_{t < l}$  represents the portion of the received sequence  $r$  before time  $l$  and  $r_{t > l}$  represents the portion of the received sequence  $r$  after time  $l$ .

So as I said we are interested in this, now this can be written as, so I have this received sequence  $r$ , so let us say this is  $r$  at time  $t = 1, t = 2$ , so this is my – let us say time instances. And I get some bits, let us say I get some  $r_1$  corresponds to what I receive at time  $t = 1, r_2$  corresponds to what I receive at time 2,  $r_1$  corresponds to what I receive in time 1 and like that,  $r_{l+1}$  is what I receive at time  $t = l+1$  like that.

So this receives this whole thing is my received sequence  $r$  okay. Now what I am doing is I am partition that received sequence into three segments. So one which corresponds to 1 is this, which corresponds to time before 1. So 1 is this portion, this portion of my received sequence, this is  $r_{t < l}$ . Next is this section which corresponds to  $r_{t > l}$  and then third section is this which corresponds to  $r_l$  okay.

So what I did was I split this  $r$  into three segment, one is  $r$  corresponds to time less than  $l, r$  at time  $l$  and  $r$  at time greater than  $l$ . Now using base rule I can write this probability as probability of  $r$  at time greater than  $l$  given  $s'$  and  $s$  and these are into probability of  $s'$   $s$  and  $r$  at  $t < l$  and  $r_l$ . Now subsequently I can further simplify this again apply base rule and I can write this as probability of  $s$  and  $r_l$  given  $s'$  and  $r_{t < l}$  into probability of  $s'$  and  $r_{t < l}$ .

So note now this term that I had here is applying base rule, essentially I broke it up into three terms, one is this term, second is this term, and third is this term okay. Now let us look at this, so probability of  $r$  when  $t > 1$  given initial state  $s'$  next state  $s$  and the received sequence before  $1$  and received sequence is  $l$ . So let us look at the trellis diagram, let us go back and look at the trellis diagram.

At time  $1$ , let us take this example of two state code, so what I had here was  $0/00$ ,  $1/11$  then I had this,  $1$  as  $10$  and this was  $0/01$ . So this was my trellis diagram, this is all zero state, this is state one okay. Now note – and like that you have – you have in trellis diagram you have – this is one trellis section, you will similarly have trellis sections are there.

So this is the time  $1$ , so you are interested in what is the probability of  $r_t$  greater than  $1$ , given previous state  $s'$ , given next state  $s$ , given the received sequence before time  $t = 1$  and given the current received sequence. Now note that if I specify this next state so probability of  $r_{t>1}$  given  $s$  then I do not need information about the previous state, I do not need information about what is the current input, I do not need information about what was the receive sequence before  $1$ , provided I know what is the next state  $s$ , so this probability that you see here, probability of  $r_{t>1}$  given  $s$  prime  $s$  and this receive sequence  $r$  can be then written as probability of  $r_{t>1}$  given only  $s$ .

Because knowing this final state  $s$  I do not need information about what was my state here, I do not information about what my receive sequence was here, I do not need information about what my past receive sequence was provided I know what was my next state  $s$  so this, given these quantity will only depend on  $s$ , so I can simplify this expression like this, now same thing here look at probability of being  $s$   $r_1$  given previous state.

And given the input before time  $t=1$  now if I specify what the pervious state is then I do not need what was my input at time  $t<1$ , so this can be simplified into this expression and then of course we have this third expression which is this, so what we have done is this join probability we have now split up into three probabilities, one is this, second one is this, and third one is this, okay and we will now show how we can compute each of these terms.



(Refer Slide Time: 20:11)

### BCJR Algorithm

- The joint pdf's  $\rho(s', s, r)$  in equation (6) can be evaluated recursively

$$\begin{aligned}
 \rho(s', s, r) &= \rho(s', s, r_{t < l}, r_l, r_{t > l}) \\
 &= \rho(r_{t > l} | s', s, r_{t < l}, r_l) \rho(s', s, r_{t < l}, r_l) \\
 &= \rho(r_{t > l} | s', s, r_{t < l}, r_l) \rho(s, r_l | s', r_{t < l}) \rho(s', r_{t < l}) \\
 &= \rho(r_{t > l} | s) \rho(s, r_l | s') \rho(s', r_{t < l}), \tag{7}
 \end{aligned}$$

where  $r_{t < l}$  represents the portion of the received sequence  $r$  before time  $l$  and  $r_{t > l}$  represents the portion of the received sequence  $r$  after time  $l$ .

(Refer Slide Time: 20:13)

## BCJR Algorithm

- Defining

$$\alpha_t(s') \equiv p(s', \mathbf{r}_{1:t}) \quad (8)$$
$$\gamma_t(s', s) \equiv p(s, \mathbf{r}_t | s') \quad (9)$$
$$\beta_{t+1}(s) \equiv p(\mathbf{r}_{t+1} | s) \quad (10)$$

(Refer Slide Time: 20:16)

### BCJR Algorithm

- Defining

$$\alpha_i(s') \equiv p(s', \mathbf{r}_{1:i}) \quad (8)$$
$$\gamma_i(s', s) \equiv p(s, \mathbf{r}_i | s') \quad (9)$$
$$\beta_{i+1}(s) \equiv p(\mathbf{r}_{i+1} | s) \quad (10)$$

So let us call this probability by  $\alpha$ , this probability by  $\gamma$ , and this probability by  $\beta$  and now we are going to show

(Refer Slide Time: 20:28)

### BCJR Algorithm

- The joint pdf's  $p(s', s, r)$  in equation (6) can be evaluated recursively

$$\begin{aligned}
 p(s', s, r) &= p(s', s, r_{t < t}, r_t, r_{t > t}) \\
 &= p(r_{t > t} | s', s, r_{t < t}, r_t) p(s', s, r_{t < t}, r_t) \\
 &= p(r_{t > t} | s', s, r_{t < t}, r_t) p(s, r_t | s', r_{t < t}) p(s', r_{t < t}) \\
 &= p(r_{t > t} | s) p(s, r_t | s') p(s', r_{t < t}), \quad (7)
 \end{aligned}$$

where  $r_{t < t}$  represents the portion of the received sequence  $r$  before time  $t$  and  $r_{t > t}$  represents the portion of the received sequence  $r$  after time  $t$ .

Then we can write then this joint probability in terms of  $\alpha$ ,  $\beta$  and  $\gamma$ .

(Refer Slide Time: 20:34)

## BCJR Algorithm

- Defining

$$\alpha_i(s') \equiv p(s', \mathbf{r}_{1:i}) \quad (8)$$
$$\gamma_i(s', s) \equiv p(s, \mathbf{r}_i | s') \quad (9)$$
$$\beta_{i+1}(s) \equiv p(\mathbf{r}_{i>1} | s) \quad (10)$$

(Refer Slide Time: 20:35)

## BCJR Algorithm

- Defining

$$\alpha_l(s') \equiv p(s', \mathbf{r}_{1:l}) \quad (8)$$

$$\gamma_l(s', s) \equiv p(s, \mathbf{r}_l | s') \quad (9)$$

$$\beta_{l+1}(s) \equiv p(\mathbf{r}_{l+1} | s), \quad (10)$$

- Equation (7) can be written as

$$p(s', s, \mathbf{r}) = \beta_{l+1}(s) \gamma_l(s', s) \alpha_l(s'). \quad (11)$$

(Refer Slide Time: 20:37)

### BCJR Algorithm

- Defining
$$\alpha_i(s') = p(s', r_{1:i}) \quad (8)$$
$$\gamma_i(s', s) \equiv p(s, r_i | s') \quad (9)$$
$$\beta_{i+1}(s) \equiv p(r_{i+1} | s), \quad (10)$$
- Equation (7) can be written as
$$p(s', s, r) = \beta_{i+1}(s) \gamma_i(s', s) \alpha_i(s'). \quad (11)$$

So we can now write our equation in terms of  $\alpha$ ,  $\beta$  and  $\gamma$ , okay, so let us now talk about how we can compute  $\alpha$ ,  $\beta$  and  $\gamma$ .

(Refer Slide Time: 20:55)

### BCJR Algorithm

- The expression for the probability  $\alpha_{l+1}(s)$  can now be rewritten as

$$\begin{aligned}\alpha_{l+1}(s) &= p(s, \mathbf{r}_{1:l+1}) = \sum_{s' \in \sigma_l} p(s', s, \mathbf{r}_{1:l+1}) \\ &= \sum_{s' \in \sigma_l} p(s, \mathbf{r}_l | s', \mathbf{r}_{1:l}) p(s', \mathbf{r}_{1:l}) \\ &= \sum_{s' \in \sigma_l} p(s, \mathbf{r}_l | s') p(s', \mathbf{r}_{1:l}) \\ &= \sum_{s' \in \sigma_l} \gamma_l(s', s) \alpha_l(s'),\end{aligned}\tag{12}$$

where  $\sigma_l$  is the set of all states at time  $l$ .

So this alpha's can be computed using forward recursion as follows.



(Refer Slide Time: 21:00)

### BCJR Algorithm

- The expression for the probability  $\alpha_{l+1}(s)$  can now be rewritten as

$$\begin{aligned}\alpha_{l+1}(s) &= p(s, \mathbf{r}_{1:l+1}) = \sum_{s' \in \sigma_l} p(s', s, \mathbf{r}_{1:l+1}) \\ &= \sum_{s' \in \sigma_l} p(s, \mathbf{r}_l | s', \mathbf{r}_{1:l}) p(s', \mathbf{r}_{1:l}) \\ &= \sum_{s' \in \sigma_l} p(s, \mathbf{r}_l | s') p(s', \mathbf{r}_{1:l}) \\ &= \sum_{s' \in \sigma_l} \gamma_l(s', s) \alpha_l(s'),\end{aligned}\tag{12}$$

where  $\sigma_l$  is the set of all states at time  $l$ .

So let us look at what is  $\alpha_{l+1}(s)$ , now go back to our definition.

(Refer Slide Time: 21:07)

### BCJR Algorithm

- Defining
$$\alpha_t(s') = p(s', r_{1:t}) \quad (8)$$
$$\gamma_t(s', s) \equiv p(s, r_t | s') \quad (9)$$
$$\beta_{t+1}(s) \equiv p(r_{t+1} | s), \quad (10)$$
- Equation (7) can be written as
$$p(s', s, r) = \beta_{t+1}(s) \gamma_t(s', s) \alpha_t(s'). \quad (11)$$

So probability, joint probability of being in state  $s$  prime and receive sequence at time  $t < 1$ .

(Refer Slide Time: 21:18)

### BCJR Algorithm

- The expression for the probability  $\alpha_{l+1}(s)$  can now be rewritten as

$$\begin{aligned}
 \alpha_{l+1}(s) &= p(s, \mathbf{r}_{1:l+1}) = \sum_{s' \in \sigma_l} \frac{p(s' | s, \mathbf{r}_{1:l+1})}{\sum_c p(a, b, c)} p(a, b) \\
 &= \sum_{s' \in \sigma_l} p(s, \mathbf{r}_l | s', \mathbf{r}_{1:l}) p(s', \mathbf{r}_{1:l}) \\
 &= \sum_{s' \in \sigma_l} p(s, \mathbf{r}_l | s') p(s', \mathbf{r}_{1:l}) \\
 &= \sum_{s' \in \sigma_l} \gamma_l(s', s) \alpha_l(s'), \tag{12}
 \end{aligned}$$

where  $\sigma_l$  is the set of all states at time  $l$ .

So  $\alpha_{l+1}$  from definition can be written like this, now I can write this as, so I am adding a new variable which is the next state  $s$  and I am adding a new variable which is next state, a previous state  $s'$  and summing over all previous state, so what is this  $\Sigma$  over  $s'$  belonging to all possible state at time  $l$ , so what I did was so I had some term probability term, probability of let us say  $(a,b)$  and what I did was I just added a term probability  $(a,b,c)$  and I summed over all possible values of  $C$  so that is what I did here, I introduced a new variable  $s$  prime and I summed over all these probabilities all these possible values of  $s$  prime, now this term can be written as product of these two terms.

(Refer Slide Time: 22:25)

## BCJR Algorithm

- Defining

$$\alpha_i(s') \equiv p(s', \mathbf{r}_{1:i}) \quad (8)$$

$$\gamma_i(s', s) \equiv p(s, \mathbf{r}_i | s') \quad (9)$$

$$\beta_{i+1}(s) \equiv p(\mathbf{r}_{i+1} | s), \quad (10)$$

- Equation (7) can be written as

$$p(s', s, \mathbf{r}) = \beta_{i+1}(s) \gamma_i(s', s) \alpha_i(s'). \quad (11)$$

(Refer Slide Time: 22:26)

### BCJR Algorithm

- Defining

$$\alpha_t(s') \equiv p(s', r_{t < t}) \quad (8)$$
$$\gamma_t(s', s) \equiv p(s, r_t | s') \quad (9)$$
$$\beta_{t+1}(s) \equiv p(r_{t > t} | s) \quad (10)$$

This is following exactly the same procedure which we followed here.

(Refer Slide Time: 22:27)

### BCJR Algorithm

- The joint pdf's  $p(s', s, r)$  in equation (6) can be evaluated recursively

$$\begin{aligned}
 p(s', s, r) &= p(s', s, r_{t<l}, r_l, r_{t>l}) \\
 &= p(r_{t>l} | s', s, r_{t<l}, r_l) p(s', s, r_{t<l}, r_l) \\
 &= p(r_{t>l} | s', s, r_{t<l}, r_l) p(s, r_l | s', r_{t<l}) p(s', r_{t<l}) \\
 &= p(r_{t>l} | s) p(s, r_l | s') p(s', r_{t<l}), \tag{7}
 \end{aligned}$$

where  $r_{t<l}$  represents the portion of the received sequence  $r$  before time  $l$  and  $r_{t>l}$  represents the portion of the received sequence  $r$  after time  $l$ .

When we wrote this we are basically using base rule.

(Refer Slide Time: 22:33)

## BCJR Algorithm

- Defining

$$\alpha_t(s') \equiv p(s', \mathbf{r}_{t < t}) \quad (8)$$
$$\gamma_t(s', s) \equiv p(s, \mathbf{r}_t | s') \quad (9)$$
$$\beta_{t+1}(s) \equiv p(\mathbf{r}_{t > t} | s), \quad (10)$$

(Refer Slide Time: 22:34)

## BCJR Algorithm

- Defining

$$\underline{\alpha}_l(s') = p(s', \mathbf{r}_{1:l}) \quad (8)$$

$$\underline{\gamma}_l(s', s) \equiv p(s, \mathbf{r}_l | s') \quad (9)$$

$$\underline{\beta}_{l+1}(s) \equiv p(\mathbf{r}_{l+1:l}), \quad (10)$$

- Equation (7) can be written as

$$p(s', s, \mathbf{r}) = \underline{\beta}_{l+1}(s) \underline{\gamma}_l(s', s) \underline{\alpha}_l(s'). \quad (11)$$



(Refer Slide Time: 22:40)

### BCJR Algorithm

- The expression for the probability  $\alpha_{l+1}(s)$  can now be rewritten as

$$\begin{aligned}
 \alpha_{l+1}(s) &= \frac{p(s, \mathbf{r}_{1:l+1})}{\sum_{s' \in \sigma_l} p(s', \mathbf{r}_{1:l+1})} = \sum_{s' \in \sigma_l} \frac{p(s, \mathbf{r}_l | s') p(s', \mathbf{r}_{1:l})}{\sum_{s' \in \sigma_l} p(s', \mathbf{r}_{1:l})} \sum_c p(a,b,c) \\
 &= \sum_{s' \in \sigma_l} \frac{p(s, \mathbf{r}_l | s') p(s', \mathbf{r}_{1:l})}{\sum_{s' \in \sigma_l} p(s', \mathbf{r}_{1:l})} \\
 &= \sum_{s' \in \sigma_l} p(s, \mathbf{r}_l | s') p(s', \mathbf{r}_{1:l}) \\
 &= \sum_{s' \in \sigma_l} \gamma_l(s', s) \alpha_l(s'), \tag{12}
 \end{aligned}$$

where  $\sigma_l$  is the set of all states at time  $l$ .

Now using base rule basically I can write this probability as product of these two probabilities, now again the probability of  $s$  and  $\mathbf{r}_l$  given  $s'$  and received sequence at time  $t < l$  if I, if you know the previous state  $s'$  you do not need this information so then this probability can be simplified to this probability and this is this, now what is this term? This term is basically by definition our  $\alpha$ .

(Refer Slide Time: 23:14)

### BCJR Algorithm

- Defining
$$\alpha_t(s') = p(s', \mathbf{r}_{1:t}) \quad (8)$$
$$\gamma_t(s', s) \equiv p(s, \mathbf{r}_t | s') \quad (9)$$
$$\beta_{t+1}(s) \equiv p(\mathbf{r}_{t+1} | s), \quad (10)$$
- Equation (7) can be written as
$$p(s', s, \mathbf{r}) = \beta_{t+1}(s) \gamma_t(s', s) \alpha_t(s'). \quad (11)$$

And what is the next term? This is our  $\gamma$ .

(Refer Slide Time: 23:20)

### BCJR Algorithm

- The expression for the probability  $\alpha_{l+1}(s)$  can now be rewritten as

$$\begin{aligned}
 \alpha_{l+1}(s) &= \frac{p(s, \mathbf{r}_{1:l+1})}{\sum_{s' \in \sigma_l} p(s', \mathbf{r}_{1:l+1})} \sum_c p(a,b,c) \\
 &= \sum_{s' \in \sigma_l} \frac{p(s, \mathbf{r}_l | s', \mathbf{r}_{1:l}) p(s', \mathbf{r}_{1:l})}{\sum_{s' \in \sigma_l} p(s, \mathbf{r}_l | s') p(s', \mathbf{r}_{1:l})} \\
 &= \sum_{s' \in \sigma_l} \gamma_l(s', s) \alpha_l(s'), \tag{12}
 \end{aligned}$$

where  $\sigma_l$  is the set of all states at time  $l$ .

So what I have shown you here then is alpha at next state  $s$  can be written as, can be computed recursively from alpha's at previous state in this particular fashion.

(Refer Slide Time: 23:36)

### BCJR Algorithm

- The expression for the probability  $\alpha_{l+1}(s)$  can now be rewritten as

$$\begin{aligned}\alpha_{l+1}(s) &= \frac{p(s, \mathbf{r}_{1:l+1})}{\sum_{s' \in \sigma_l} p(s', \mathbf{r}_{1:l+1})} \sum_c p(a,b,c) \\ &= \sum_{s' \in \sigma_l} \frac{p(s, \mathbf{r}_l | s', \mathbf{r}_{1:l}) p(s', \mathbf{r}_{1:l})}{\sum_{s' \in \sigma_l} p(s, \mathbf{r}_l | s', \mathbf{r}_{1:l}) p(s', \mathbf{r}_{1:l})} \\ &= \sum_{s' \in \sigma_l} \gamma_l(s', s) \alpha_l(s'),\end{aligned}\quad (12)$$

where  $\sigma_l$  is the set of all states at time  $l$ .

So again let us illustrate this with an example, let us go back to our

(Refer Slide Time: 23:40)

**BCJR Algorithm**

• The expression for the probability  $\alpha_{l+1}(s)$  can now be rewritten as

$$\alpha_{l+1}(s) = \frac{p(s, r_{l+1})}{\sum_{s' \in \sigma_l} p(s' | s, r_{l+1})} = \sum_{s' \in \sigma_l} \frac{p(s, r_l | s') p(s', r_{l+1})}{\sum_{s'' \in \sigma_l} p(s, r_l | s'') p(s'', r_{l+1})} p(a, b)$$

$$\alpha_{l+1}(0) = \gamma_{\ell}(0, 0) \alpha_{\ell}(0) + \gamma_{\ell}(1, 0) \alpha_{\ell}(1)$$

$$\alpha_{l+1}(s) = \sum_{s' \in \sigma_l} \frac{p(s, r_l | s') p(s', r_{l+1})}{\sum_{s'' \in \sigma_l} p(s, r_l | s'') p(s'', r_{l+1})} p(a, b)$$

$$\alpha_{l+1}(s) = \sum_{s' \in \sigma_l} \gamma_{\ell}(s', s) \alpha_{\ell}(s')$$

where  $\sigma_l$  is the set of all states at time  $l$ .

Two state code example, so this is two state code, this is my all zero state, this is state 1, so the two transition let us say this is zero input, output 00, input 1, output 11, here input 1, output 10 and here input 0, and output 01, so then what would be the value of alpha so let us say this is time equal to some  $l$  and this is time  $t = l+1$  so how can we write let us say alpha at  $l+1$  for the state zero, now note here this is given by product of this  $\Sigma$  over all input state, right?

Now so  $\alpha_l^0$  can be written as, then gamma this can be written as  $\gamma_l(0, 0)$   $\gamma(0, 0)$  is previous state zero next state is zero,  $\gamma(0, 0)$  into  $\alpha_l$  belonging to state zero so this is  $\gamma_l(0, 0)$   $\alpha_l(0)$  so this is corresponding to this transition, okay this is corresponding to this transition this term will come, fine now there is another transition here which is called basically this so we can write this will be plus  $\gamma_l(1, 0)$  so  $\gamma_l(1, 0)$  is a gamma corresponding to this transition when the initial state is one and next state is zero.

Multiplied by  $\alpha_l(1)$  okay so  $\alpha_{l+1}(0)$  can be then written as this, now similarly we can also compute what is the value of  $\alpha_{l+1}(1)$  so we repeat the same procedure so let us write it here  $\alpha_{l+1}$  when the final state is one can be written as  $\gamma_l(0, 1) \alpha_l(0)$  plus so this is corresponding to

this transition gamma L initial state zero, final state one and alpha at time L zero plus this another transition which is this.

So this can be written as  $\gamma_1(1, 1) \alpha_1(1)$ , so this these are for this particular convolutional encoder whose trellis section is given by this, these are these two are my alpha values, this one and this.

So you can see I can recursively compute alpha at time l+1 from alpha at time l and branch metrics gamma, now to do this recursion we need to know what is the initial condition, so what is the initial condition?

(Refer Slide Time: 27:44)

**BCJR Algorithm**

• The expression for the probability  $\alpha_{l+1}(s)$  can now be rewritten as

$$\alpha_{l+1}(s) = p(s, r_{l+1}) = \sum_{s' \in \sigma_l} p(s' | s, r_{l+1}) \sum_c p(a, b, c)$$

$$= \sum_{s' \in \sigma_l} p(s, r_l | s', r_{l+1}) p(s', r_{l+1})$$

$$= \sum_{s' \in \sigma_l} p(s, r_l | s') p(s', r_{l+1})$$

$$= \sum_{s' \in \sigma_l} \gamma_l(s', s) \alpha_l(s')$$

where  $\sigma_l$  is the set of all states at time  $l$ .

Handwritten notes and diagram:

- A red box on the left contains the equation:  $\alpha_{l+1}(0) = \gamma_l(0,0)\alpha_l(0) + \gamma_l(1,0)\alpha_l(1)$
- A red box on the right contains the equation:  $\alpha_{l+1}(1) = \gamma_l(0,1)\alpha_l(0) + \gamma_l(1,1)\alpha_l(1)$
- A trellis diagram shows two states at time  $t=l$  (0 and 1) and two states at time  $t=l+1$  (0 and 1). Transitions are labeled with bits: 0/01, 1/10, 0/00, 1/11.
- Red text next to the diagram asks:  $\alpha_0(0) = ?$  and  $\alpha_0(1) = ?$

So we need to know what is the value of  $\alpha_0$  for different states, for state zero, for state one, we need to know what the value of these are. Now note initially we assume that the encoder is in all zero state so if you assume the encoder is in all zero state.

(Refer Slide Time: 28:09)

### BCJR Algorithm

• The expression for the probability  $\alpha_{l+1}(s)$  can now be rewritten as

$$\alpha_{l+1}(s) = \frac{p(s, r_{l+1})}{\sum_{s' \in \sigma_l} p(s', r_{l+1})} = \sum_{c \in \mathcal{C}} \frac{p(s, r_{l+1} | s, c)}{\sum_{s' \in \sigma_l} \sum_{c \in \mathcal{C}} p(s', r_{l+1} | s', c)} p(s, r_{l+1} | s, c)$$

$$= \sum_{s' \in \sigma_l} \frac{p(s, r_l | s') p(s', r_{l+1} | s, c)}{\sum_{s'' \in \sigma_l} p(s, r_l | s'') p(s'', r_{l+1} | s, c)} p(s, r_{l+1} | s, c)$$

$$= \sum_{s' \in \sigma_l} \gamma_l(s', s) \alpha_l(s')$$

where  $\sigma_l$  is the set of all states at time  $l$ .

$$\alpha_{l+1}(0) = \gamma_l(0,0)\alpha_l(0) + \gamma_l(1,0)\alpha_l(1)$$

$$\alpha_{sH}(i) = \gamma_r(a_i)\alpha_x(b_i) + \gamma_r(v_i)\alpha_x(i)$$

Then it is, it is going to stay in all zero state then in that case we consider this probability as one and this all other possibility of it staying in all the state as zero, so the initial value when we assume that the encoder is in all zero state we assume that alpha zero at zero is one and at alpha zero at any other state is non zero.

(Refer Slide Time: 28:34)

### BCJR Algorithm

- Similarly expression of the probability  $\beta_i(s')$  can be written as

$$\begin{aligned}\beta_i(s') &= p(\mathbf{r}_{t>(i-1)}|s') & (13) \\ &= \sum_{s \in \sigma_{i+1}} p(\mathbf{r}_{t>(i-1)}, s|s') \\ &= \sum_{s \in \sigma_{i+1}} p(\mathbf{r}_{t>i}, \mathbf{r}_i, s|s') \\ &= \sum_{s \in \sigma_{i+1}} p(\mathbf{r}_{t>i}|s', s, \mathbf{r}_i) p(s, \mathbf{r}_i|s') \\ &= \sum_{s \in \sigma_{i+1}} p(\mathbf{r}_{t>i}|s) p(s, \mathbf{r}_i|s') \\ &= \sum_{s \in \sigma_{i+1}} \beta_{i+1}(s) \gamma_i(s', s)\end{aligned}$$

where  $\sigma_{i+1}$  is the set of all states at time  $i+1$ .

So similarly we can compute betas recursively so from definition we can write betas in this particular fashion.



(Refer Slide Time: 28:40)

**BCJR Algorithm**

• Similarly expression of the probability  $\beta_l(s')$  can be written as

$$\beta_l(0) = \beta_{2H}(0)\gamma_l(0,1) + \beta_{2H}(0)\gamma_l(0,0)$$

$$\beta_l(1) = \beta_{2H}(0)\gamma_l(1,0) + \beta_{2H}(1)\gamma_l(1,1)$$

$$\begin{aligned} \beta_l(s') &= \frac{p(r_{t>(l-1)}|s')}{(13)} \\ &= \sum_{s \in \sigma_{l+1}} p(r_{t>(l-1)}|s, s') \\ &= \sum_{s \in \sigma_{l+1}} p(r_{t>l}, r_l, s|s') \\ &= \sum_{s \in \sigma_{l+1}} p(r_{t>l}|s', s) p(r_l|s, r_l, s') \\ &= \sum_{s \in \sigma_{l+1}} p(r_{t>l}|s) p(s, r_l|s') \\ &= \sum_{s \in \sigma_{l+1}} \beta_{l+1}(s) \gamma_l(s', s) \end{aligned}$$

where  $\sigma_{l+1}$  is the set of all states at time  $l+1$ .

And again we introduced a new variable  $s$  and sum over all possible values of  $s$ , so then this becomes from here we get this, now we split this  $r$  into these two terms so we get this expression, now using base rule I can separate out this term into two terms like this and we know that, again let us go back to our trellis section so 0/00, 1/11 this is 1/10 and this is 0/01, this is state 0 this state 1.

So if you are interested in probability  $r_{l+1}$  that is  $r$ , this is your time= $l$  so probability of  $r_{l+1}$  given previous state  $s'$  next state  $s$  and  $r_l$  it only depends on, so if you know the next state  $s$  you do not need information about the previous state, you do not need information about the current bits. So I can simplify this expression in this particular fashion and if we go back this is nothing but our  $\beta$  and this is our  $\gamma$ , so let us compute  $\beta$  for this particular code, so we are interested in computing  $\beta_l(0)$  and  $\beta_l(1)$  so  $\beta_l(0)$  would be so  $\beta_l(0)$

So we are interested in computing  $\beta(0)$  so this is sum over all those transitions which are ending at this state, so there are two transitions, one is this one another is this one, so let us write the expression for this particular term, this we can write as  $\beta_{l+1}(1)$  times  $\gamma_l(0,1)$  so the contribution of this is  $\beta_{l+1}$  corresponding to state 1 multiplied by  $\gamma$  of this trellis section  $\gamma_l$  when the initial state is

zero and the next state is 1 so this, this will contribute this term, now plus is another transition which is this, this one right? So we can write contribution of this as  $\beta_{l+1}(0)$  times  $\gamma_l(0,0)$  so this is our expression of  $\beta_l$  for state zero, similarly we can compute  $\beta_l$  for state one, so what are the two transitions which are ending at this state, one is this one, other one is this one.

So let us write down the expression for this one this one, so this will be  $\beta_{l+1}(0)$  times  $\gamma_l(1,0)$  this is this term and what about this particular term, this will be given by  $\beta_{l+1}(1)$  times  $\gamma_l(1,1)$  so this is our expression for  $\beta$  so as you can see similar to the expression for  $\alpha$ , now these  $\beta$  can be computed using so  $\alpha$  can be computed using forward recursion and similarly  $\beta$  can be computed using backward recursion.

So we would require the knowledge of  $\beta$  at time at end of the trellis, now how do we know the values of  $\beta$ , now if then encoder is terminated that means if the encoder is brought back to all zero state in that case  $\beta$  at end of the trellis at end of the time let us say.

(Refer Slide Time: 33:59)

**BCJR Algorithm**

• Similarly expression or the probability  $\beta_l(s')$  can be written as  $\beta_k(0) = 1$   
 $\beta_k(1) = 0$   
 (13)

$$\beta_l(s') = \frac{p(r_{l+1}|s')}{\sum_{s \in \sigma_{l+1}} p(r_{l+1}|s, s')} = \sum_{s \in \sigma_{l+1}} \frac{p(r_{l+1}|s, s') p(s, r_l | s')}{\sum_{s \in \sigma_{l+1}} p(r_{l+1}|s, s') p(s, r_l | s')} = \sum_{s \in \sigma_{l+1}} \beta_{l+1}(s) \gamma_l(s', s)$$

where  $\sigma_{l+1}$  is the set of all states at time  $l+1$ .

Let us call  $\beta$  at time  $k$  which is the end of the block at state zero will be one and for all other state it will be for in this case there are only two states so for all other state this will be zero, this is for

the case when the convolutional encoder is brought back to all zero state, it is terminated. In case the convolutional encoder is not terminated then we do not know in which state it has ended up with so what we will do is in that case we will assume that it is equally likely to end up at all zero state or any other state.

(Refer Slide Time: 34:47)

**BCJR Algorithm**

• Similarly expression of the probability  $\beta_l(s')$  can be written as  $\beta_k(0)=1$   
 $\beta_k(1)=0$  (13)

$$\beta_l(s') = \frac{p(\mathbf{r}_{l>l-1}|s')}{\sum_{s \in \sigma_{l+1}} p(\mathbf{r}_{l>l-1}, s|s')} = \sum_{s \in \sigma_{l+1}} \frac{p(\mathbf{r}_{l>l-1}|s') \cdot p(s, r_l|s')}{\sum_{s \in \sigma_{l+1}} p(\mathbf{r}_{l>l-1}|s') \cdot p(s, r_l|s')} = \sum_{s \in \sigma_{l+1}} \beta_{l+1}(s) \gamma_l(s', s)$$

where  $\sigma_{l+1}$  is the set of all states at time  $l+1$ .

So in that case we would assume  $\beta$  at the end of the block to be equal to 1 by number of states so in this case in we would assume that  $\beta_k(0) = 1/2$  and  $\beta_k(1) = 1/2$  so this is for the case when convolutional encoder is not terminated, that mean it is not brought back to all zero state and this will be the initial condition when the convolutional encoder is terminated.

(Refer Slide Time: 35:16)

### BCJR Algorithm

- The branch metric  $\gamma_l(s', s)$  can be written as

$$\begin{aligned}\gamma_l(s', s) &= p(s, \mathbf{r}_l | s') = \frac{p(s', s, \mathbf{r}_l)}{P(s')} & (14) \\ &= \left[ \frac{P(s', s)}{P(s')} \right] \left[ \frac{p(s', s, \mathbf{r}_l)}{P(s', s)} \right] \\ &= P(s|s') p(\mathbf{r}_l | s', s) = P(u_l) p(\mathbf{r}_l | \mathbf{v}_l),\end{aligned}$$

where  $u_l$  is the input bit and  $\mathbf{v}_l$  the output bits corresponding to the state transition  $s' \rightarrow s$  at time  $l$ .

Now next we compute the branch metrics gamma.

(Refer Slide Time: 35:21)

The screenshot shows a presentation slide with a red header containing the text "BCJR Algorithm". Below the header, there is a bullet point: "• The branch metric  $\gamma_l(s', s)$  can be written as". This is followed by a mathematical derivation labeled (14):  
$$\begin{aligned}\gamma_l(s', s) &= \frac{p(s, r_l | s')}{P(s')} = \frac{p(s', s, r_l)}{P(s')} \\ &= \frac{P(s', s)}{P(s')} \left[ \frac{p(s', s, r_l)}{P(s', s)} \right] \\ &= P(s|s') p(r_l | s', s) = P(u_l) p(r_l | v_l),\end{aligned}\tag{14}$$

where  $u_l$  is the input bit and  $v_l$  the output bits corresponding to the state transition  $s' \rightarrow s$  at time  $l$ .

Now from definition gammas can be written like this so this can be written as joint probability of being in previous state  $s$  prime next state  $s$  given a receive sequence at time  $l$ ,  $r_l/P(s')$  prime. Now this, I introduce a term this, so I add this term into numerator, I similarly add this term in the denominator okay now this, this quantity can be written as  $P(s|s')$  and this probability can be written as probability of  $r_l$ , given previous state  $s$  prime and next state  $S$  which can also be written as probability of  $r_l$  given transmit sequence  $V_l$  multiplied by a-priori probability of getting  $U_l$ , so note that this probability will be one only when there is a valid transition from state  $S'$  prime to  $S$  otherwise this will be zero okay.

(Refer Slide Time: 36:39)

The slide is titled "BCJR Algorithm" and contains the following text and equations:

- The branch metric  $\gamma_l(s', s)$  can be written as

$$\begin{aligned}\gamma_l(s', s) &= \frac{p(s, \mathbf{r}_l | s')}{P(s')} = \frac{p(s', s, \mathbf{r}_l)}{P(s')} & (14) \\ &= \frac{P(s', s)}{P(s')} \frac{p(s', s, \mathbf{r}_l)}{P(s', s)} \\ &= \frac{P(s | s') p(\mathbf{r}_l | s', s)}{P(s | s') p(\mathbf{r}_l | s', s)} = P(u_l) p(\mathbf{r}_l | \mathbf{v}_l),\end{aligned}$$

where  $u_l$  is the input bit and  $\mathbf{v}_l$  the output bits corresponding to the state transition  $s' \rightarrow s$  at time  $l$ .

So what does gamma depends on, it depends on what is the a-priori probability of  $U_l$  and it depends on this likelihood function probability of  $\mathbf{r}_l$  given  $\mathbf{V}_L$ .

(Refer Slide Time: 36:51)

**BCJR Algorithm**

**BPSK**

- For a continuous output AWGN channel, if  $s' \rightarrow s$  is a valid state transition,

$$\gamma_i(s', s) = P(u_i) p(r_i | v_i) = P(u_i) \left( \sqrt{\frac{E_s}{\pi N_0}} \right)^n e^{-\frac{E_s}{N_0} \|r_i - v_i\|^2} \quad (15)$$

where  $\|r_i - v_i\|^2$  is the squared Euclidean distance between the (normalized by  $\sqrt{E_s}$ ) received branch  $r_i$  and the transmitted branch  $v_i$  at time  $i$ .

*Handwritten notes on the slide:*  
 - A blue box highlights  $P(u_i)$ .  
 - A red box highlights  $\left( \sqrt{\frac{E_s}{\pi N_0}} \right)^n$ .  
 - A red box highlights  $e^{-\frac{E_s}{N_0} \|r_i - v_i\|^2}$ .  
 - Red annotations show the expansion of the squared distance:  $\|r_i - v_i\|^2 = r_i^2 + v_i^2 - 2r_i v_i$ .

Now if I consider an additive wide Gaussian noise channel we can write this probability of  $R_L$  given  $V$  in this particular fashion so gamma for an A w channel will then be given by this expression, so note this depends on a-priori probability of  $U_i$ , it depends on the Euclidean distance between  $R_i$  and  $V_i$ . Now let us assume that we are considering a binary phase shift key so in another words basically we have bits map to plus one and minus one let us say or plus, plus ES and - ES okay, so let us expand

This term and see can we simplify this term, now this term is, this term will be common for all the terms which is, which depends only on signal to noise ratio and if you look at this particular term so here there is an  $R_i^2$  term, there is  $V_i^2$  term, and then there is  $- 2 R_i V_i$  term, so this  $R_i, r_i^2$  term that does not depend on what my VL is, and since we are considering a BPSK modulate signal so  $V_i$  whether UFL is  $- 1$  or  $+ 1$  this will basically be the same.

(Refer Slide Time: 38:38)

**BPSK**

- For a continuous output AWGN channel, if  $s' \rightarrow s$  is a valid state transition,

$$\gamma_l(s', s) = P(u_l) p(r_l | v_l) = P(u_l) \left( \sqrt{\frac{E_s}{\pi N_0}} \right)^n e^{-\frac{E_s}{N_0} \|r_l - v_l\|^2} \quad (15)$$

where  $\|r_l - v_l\|^2$  is the squared Euclidean distance between the (normalized by  $\sqrt{E_s}$ ) received branch  $r_l$  and the transmitted branch  $v_l$  at time  $l$ .

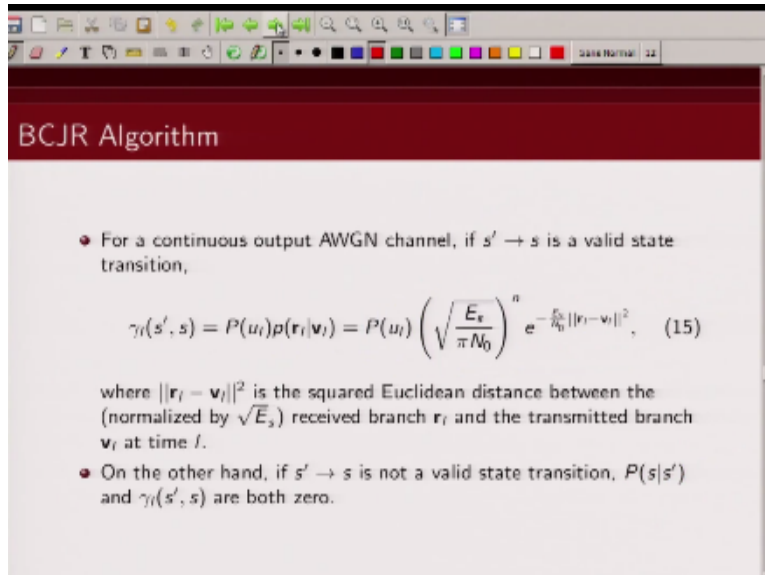
$\gamma_l^2 + v_l^2 - 2r_l v_l$   
 $\gamma_l(s', s) = \frac{E_s}{N_0} \frac{r_l v_l}{4}$

This will be just one, so the only term that is changing the choice of  $V_1$  is this particular term, so what we can simplify this gamma L, we can just simplify our gamma L like this, so it is basically probability of  $U_L$  and exponent  $-E_s / N_0$  and this is basically two times  $R_1 V_1$ , so it becomes dot product between the receive sequence and this transmit code word  $V_1$  okay so, and of course there is some constant, there is some constant term  $K_1$  which is common, so in nutshell then our gamma depends on this term right. And it depends on what the initial a-priori probability of  $U_L$  is, so for an additive wide Gaussian noise channel when we are implying BPSK modulation then

We can simplify our expression for gamma so this can be written as  $E$  [indiscernible][00:39:53] two times  $R_1 \cdot V_1$  now what is this  $R_1 \cdot V_1$ , we will illustrate this with an example when we solve when we show an example okay, so the point which I am trying to make is that this expression that you see for computation of gamma for an additive wide Gaussian noise it essentially depends on 2 terms, one is this and another is this term.



(Refer Slide Time: 40:25)



The image shows a presentation slide with a dark red header containing the text "BCJR Algorithm". Below the header, there is a list of bullet points and a mathematical equation. The first bullet point states: "For a continuous output AWGN channel, if  $s' \rightarrow s$  is a valid state transition,". This is followed by the equation 
$$\gamma_l(s', s) = P(u_l)p(\mathbf{r}_l|\mathbf{v}_l) = P(u_l) \left( \sqrt{\frac{E_s}{\pi N_0}} \right)^n e^{-\frac{E_s}{N_0} \|\mathbf{r}_l - \mathbf{v}_l\|^2}, \quad (15)$$
 where  $\|\mathbf{r}_l - \mathbf{v}_l\|^2$  is the squared Euclidean distance between the (normalized by  $\sqrt{E_s}$ ) received branch  $\mathbf{r}_l$  and the transmitted branch  $\mathbf{v}_l$  at time  $l$ . The second bullet point states: "On the other hand, if  $s' \rightarrow s$  is not a valid state transition,  $P(s|s')$  and  $\gamma_l(s', s)$  are both zero."

BCJR Algorithm

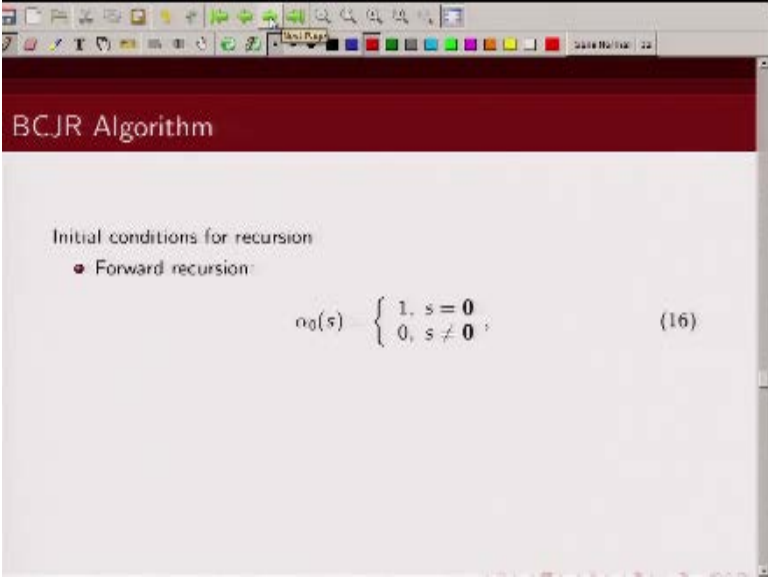
- For a continuous output AWGN channel, if  $s' \rightarrow s$  is a valid state transition,

$$\gamma_l(s', s) = P(u_l)p(\mathbf{r}_l|\mathbf{v}_l) = P(u_l) \left( \sqrt{\frac{E_s}{\pi N_0}} \right)^n e^{-\frac{E_s}{N_0} \|\mathbf{r}_l - \mathbf{v}_l\|^2}, \quad (15)$$

where  $\|\mathbf{r}_l - \mathbf{v}_l\|^2$  is the squared Euclidean distance between the (normalized by  $\sqrt{E_s}$ ) received branch  $\mathbf{r}_l$  and the transmitted branch  $\mathbf{v}_l$  at time  $l$ .

- On the other hand, if  $s' \rightarrow s$  is not a valid state transition,  $P(s|s')$  and  $\gamma_l(s', s)$  are both zero.

(Refer Slide Time: 40:26)



The image shows a screenshot of a presentation slide. The slide has a dark red header with the text "BCJR Algorithm" in white. Below the header, the text "Initial conditions for recursion" is displayed. Underneath, there is a bullet point labeled "Forward recursion:" followed by a piecewise function for  $\alpha_0(s)$ . The function is defined as 1 when  $s = \mathbf{0}$  and 0 when  $s \neq \mathbf{0}$ . The equation is labeled (16) on the right side.

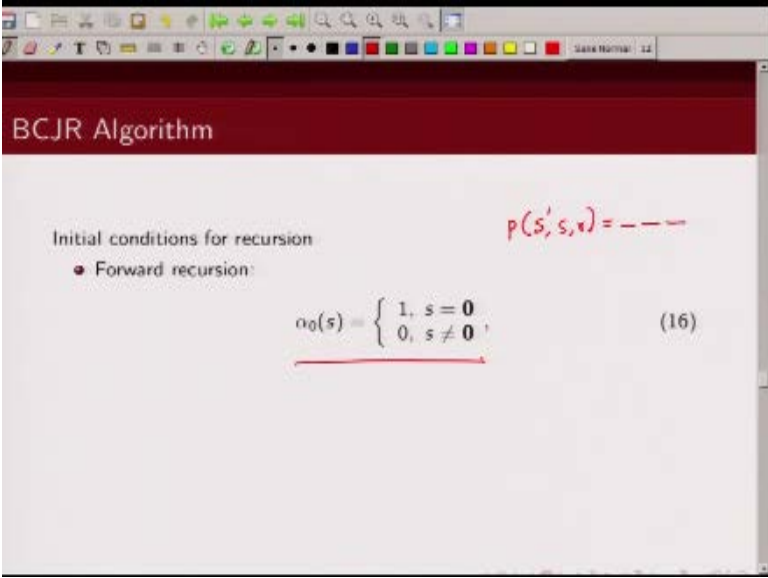
Initial conditions for recursion

- Forward recursion:

$$\alpha_0(s) = \begin{cases} 1, & s = \mathbf{0} \\ 0, & s \neq \mathbf{0} \end{cases} \quad (16)$$

Next I have already specified now that our joint probability of

(Refer Slide Time: 40:35)



The image shows a presentation slide titled "BCJR Algorithm". The slide content includes:

- Initial conditions for recursion
  - Forward recursion:

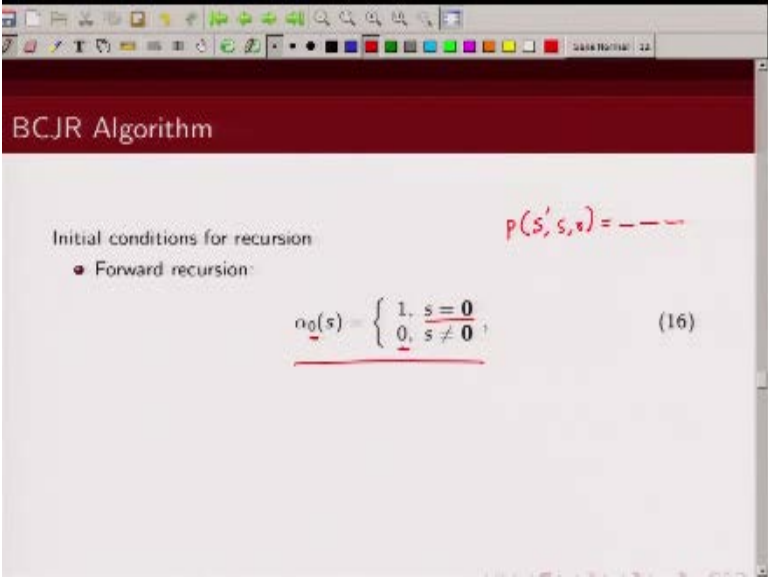
Handwritten notes in red ink are present:

- $p(s', s, v) = \dots$  (partially visible)
- $\alpha_0(s) = \begin{cases} 1, & s = \mathbf{0} \\ 0, & s \neq \mathbf{0} \end{cases}$  (Equation 16)

The equation for  $\alpha_0(s)$  is underlined in red.

The joint probability that we compute it, it is basically a product of 3 terms alpha, beta and gamma, now alpha, beta can be computed in a recursive fashion and I already mentioned that usually our encoder is in all zero state to start off with and that is why we assume.

(Refer Slide Time: 40:55)



The image shows a presentation slide titled "BCJR Algorithm". The slide content includes:

- Initial conditions for recursion
  - Forward recursion:

The equation for the initial condition is:

$$\alpha_0(s) = \begin{cases} 1, & s = \mathbf{0} \\ 0, & s \neq \mathbf{0} \end{cases} \quad (16)$$

There is a handwritten red note in the top right corner:  $p(s', s, v) = \dots$

That alpha at time zero is one for the state zero and it is zero for all other state.

(Refer Slide Time: 41:10)

BCJR Algorithm

Initial conditions for recursion

- Forward recursion:  
$$\alpha_0(s) = \begin{cases} 1, & s = 0 \\ 0, & s \neq 0 \end{cases} \quad (16)$$
- Backward recursion:  
$$\beta_k(s) = \begin{cases} 1, & s = 0 \\ 0, & s \neq 0 \end{cases} \quad (17)$$

Similarly if we assume that our encoder is terminated, that means it has been brought back to all zero state, in that case at the end of the block which is our  $k$ ,  $b_k$  will be one for state zero, and zero for all other states. So these are our initial condition for computing the recursion for forward recursion as well as backward recursion, so now then.

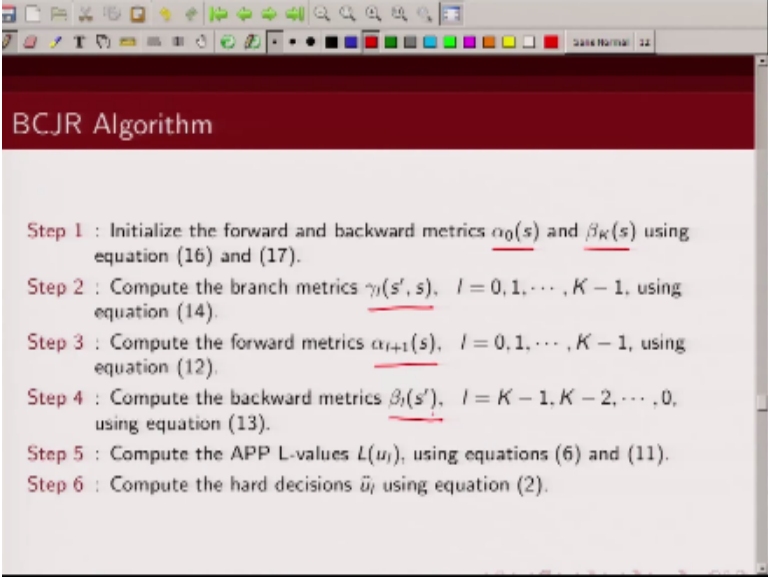
(Refer Slide Time: 41:41)

**BCJR Algorithm**

- Step 1** : Initialize the forward and backward metrics  $\alpha_0(s)$  and  $\beta_K(s)$  using equation (16) and (17).
- Step 2** : Compute the branch metrics  $\gamma_l(s', s)$ ,  $l = 0, 1, \dots, K - 1$ , using equation (14).
- Step 3** : Compute the forward metrics  $\alpha_{l+1}(s)$ ,  $l = 0, 1, \dots, K - 1$ , using equation (12).
- Step 4** : Compute the backward metrics  $\beta_l(s')$ ,  $l = K - 1, K - 2, \dots, 0$ , using equation (13).
- Step 5** : Compute the APP L-values  $L(u_i)$ , using equations (6) and (11).
- Step 6** : Compute the hard decisions  $\hat{u}_i$  using equation (2).

To recap how do we compute the a-posteriori probability, the first thing is we need to initialize the values of alpha at time zero and beta at time end of the block which is I am calling K. The next thing I need to do is now to compute alpha and beta I need the value of these branch metrics gamma, so the first thing I need to do is I need to compute this branch metrics gamma, so I will compute this branch metric for all valid transitions and for all time instances so that is.

(Refer Slide Time: 42:22)

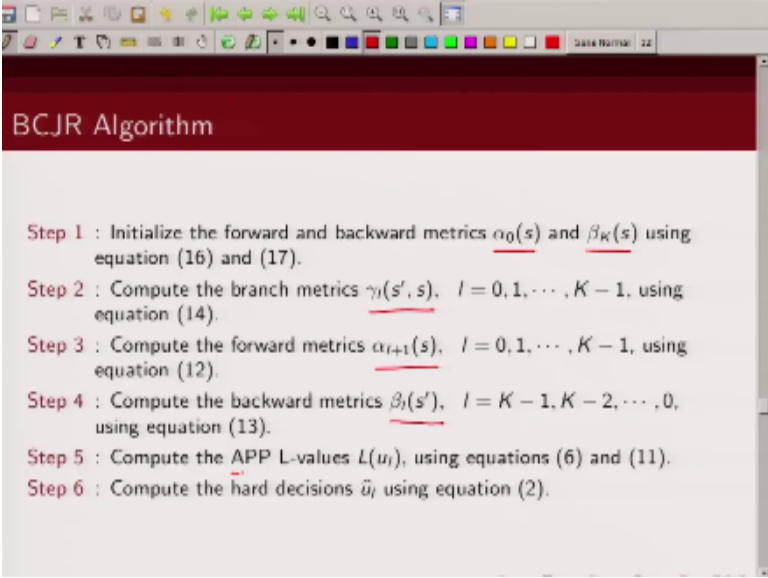


The image shows a presentation slide with a dark red header containing the text "BCJR Algorithm". Below the header, there are six numbered steps, each with a red underline under a specific mathematical term. The steps are as follows:

- Step 1 : Initialize the forward and backward metrics  $\alpha_0(s)$  and  $\beta_K(s)$  using equation (16) and (17).
- Step 2 : Compute the branch metrics  $\gamma_l(s', s)$ ,  $l = 0, 1, \dots, K - 1$ , using equation (14).
- Step 3 : Compute the forward metrics  $\alpha_{l+1}(s)$ ,  $l = 0, 1, \dots, K - 1$ , using equation (12).
- Step 4 : Compute the backward metrics  $\beta_l(s')$ ,  $l = K - 1, K - 2, \dots, 0$ , using equation (13).
- Step 5 : Compute the APP L-values  $L(u_i)$ , using equations (6) and (11).
- Step 6 : Compute the hard decisions  $\hat{u}_i$  using equation (2).

The second step, the third step is once I compute this branch metrics gamma then I will compute using forward recursion I will compute the values of alphas and using backward recursion I will compute the values of beta. Once I have the values of alpha, beta and gamma then I can compute.

(Refer Slide Time: 42:46)



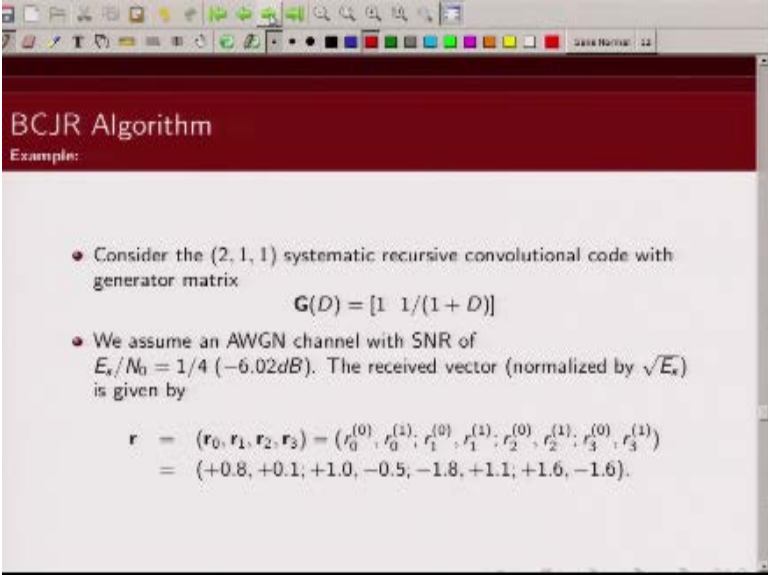
The image shows a presentation slide titled "BCJR Algorithm" with a dark red header. The slide contains six numbered steps describing the algorithm's process. The text is as follows:

- Step 1 : Initialize the forward and backward metrics  $\alpha_0(s)$  and  $\beta_K(s)$  using equation (16) and (17).
- Step 2 : Compute the branch metrics  $\gamma_l(s', s)$ ,  $l = 0, 1, \dots, K - 1$ , using equation (14).
- Step 3 : Compute the forward metrics  $\alpha_{l+1}(s)$ ,  $l = 0, 1, \dots, K - 1$ , using equation (12).
- Step 4 : Compute the backward metrics  $\beta_l(s')$ ,  $l = K - 1, K - 2, \dots, 0$ , using equation (13).
- Step 5 : Compute the APP L-values  $L(u_l)$ , using equations (6) and (11).
- Step 6 : Compute the hard decisions  $\hat{u}_l$  using equation (2).

The a-posteriori probability because I, I have shown that it is a basically product of these three terms so I can then.



(Refer Slide Time: 42:55)



**BCJR Algorithm**  
Example:

- Consider the (2, 1, 1) systematic recursive convolutional code with generator matrix  
$$\mathbf{G}(D) = [1 \quad 1/(1 + D)]$$
- We assume an AWGN channel with SNR of  $E_s/N_0 = 1/4$  (-6.02dB). The received vector (normalized by  $\sqrt{E_s}$ ) is given by

$$\begin{aligned} \mathbf{r} &= (\mathbf{r}_0, \mathbf{r}_1, \mathbf{r}_2, \mathbf{r}_3) = (r_0^{(0)}, r_0^{(1)}, r_1^{(0)}, r_1^{(1)}, r_2^{(0)}, r_2^{(1)}, r_3^{(0)}, r_3^{(1)}) \\ &= (+0.8, +0.1, +1.0, -0.5, -1.8, +1.1, +1.6, -1.6). \end{aligned}$$

Compute this APP values and once I have this APP value I will take a hard decision based on whether this is greater than zero or plus one so the final thing that I going to do is I am going to take a hard decision based on what is the value of this APP value okay, so let us now show this.

(Refer Slide Time: 43:23)

**BCJR Algorithm**  
Example:

- Consider the  $(2, 1, 1)$  systematic recursive convolutional code with generator matrix  
$$\mathbf{G}(D) = [1 \quad 1/(1+D)]$$
- We assume an AWGN channel with SNR of  $E_s/N_0 = 1/4$  ( $-6.02\text{dB}$ ). The received vector (normalized by  $\sqrt{E_s}$ ) is given by  
$$\mathbf{r} = (r_0, r_1, r_2, r_3) = (r_0^{(0)}, r_0^{(1)}, r_1^{(0)}, r_1^{(1)}, r_2^{(0)}, r_2^{(1)}, r_3^{(0)}, r_3^{(1)})$$
$$= (+0.8, +0.1; +1.0, -0.5; -1.8, +1.1; +1.6, -1.6).$$

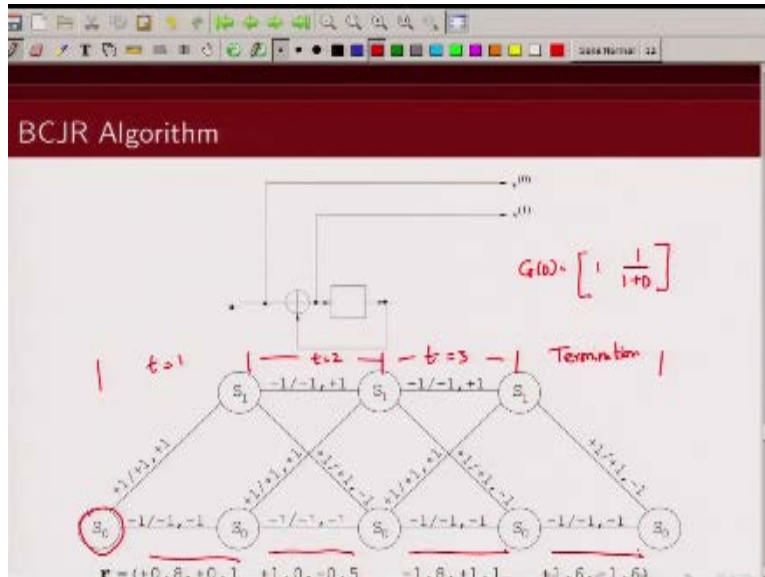
*Handwritten notes:* BPSK,  $P(u) = \begin{cases} 1/2 & u=1 \\ -1/2 & u=-1 \end{cases}$

The same using an example, so we are going to consider an example to illustrate how we can do BCJR decoding, so we are considering our rate  $1/2$  convolutional code with memory one whose generator sequence is basically given by this, the generator matrix is given by this. We are considering BPSK modulation and we are assuming that initial probability  $U_1$  is equally likely to be plus 1 or minus 1 so we are assuming that it is equally likely to be it is plus 1 with probability half and minus 1 with probability half.

We are considering an AWGN channel with SNR of  $1/4$  and we are assuming that the receive signal are normalized by under route DFS so what we are receiving is this particular sequence. The question I am interested is if the receive sequence is this I am interested in estimating what was my information sequence. So to solve this problem what we need to do is, we need to compute the a-posteriori L value. Now to compute the a-posteriori L value we will first have to compute alpha, beta, and gammas okay.

And eventually we will compute the a-posteriori L value and then I will take a hard decision on that to decide, estimate our information sequence, so this is the.

(Refer Slide Time: 45:09)

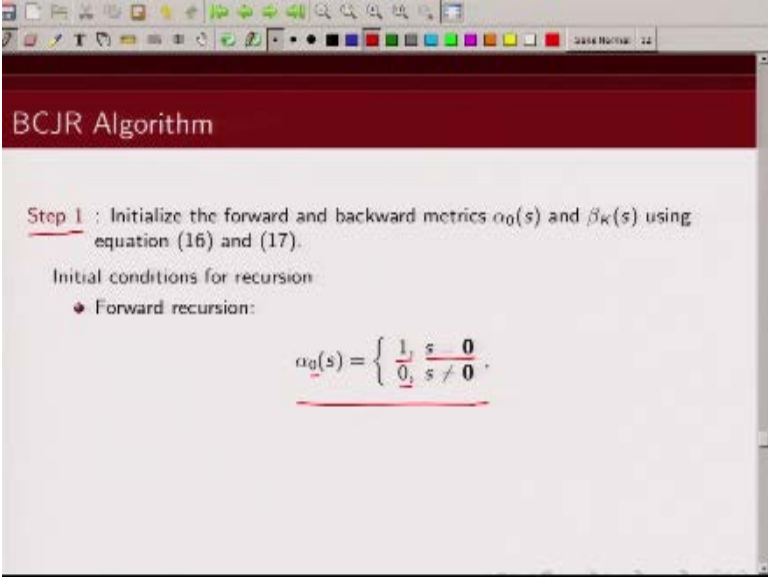


Convolutional encoder that we have consider, this is basically  $G(D)$  is the rate  $\frac{1}{2}$  code and its corresponding trellis diagram is this. For simplicity I have just considered 4 time instances so initially I assumed this encoder is in all zero state which is denoted by  $S_0$  and it gets some bits, it moves to either  $S_0$  and  $S_1$  depending on what bits it has received, this is the first time instance, this is first time instance, this is  $T=2$ , this is  $T=3$ , and then after this what I am doing is I am terminating this encoder back to all zero state.

So this is the termination phase, so I bring this encoder back to all zero state. Now this is a rate  $\frac{1}{2}$  code of our each trellis section I am receiving 2 bits so at time  $T=1$  what I received is these 2 bits, point 8 and plus point 1, for  $T=2$  I received these 2 plus point 1 and minus point 5, for  $T=3$  I received minus 1.8 and plus 1.1, and for during the termination phase I received plus 1.6 and minus 1.6. Please note I am interested in, given this receive sequence I am interested in estimating what was the information bit.

That was transmitted at time  $T=1$ , what was the information bit that was transmitted at time  $T=2$  what was the information bit that was transmitted at time  $T=3$ .

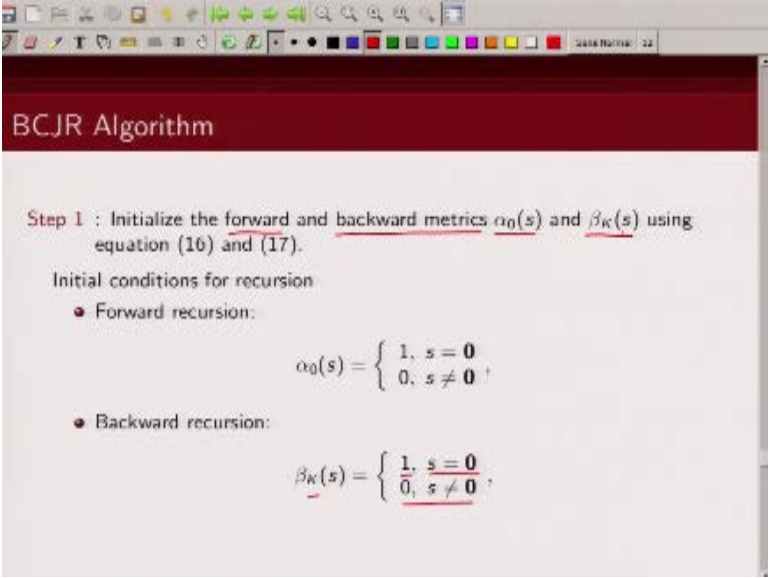
(Refer Slide Time: 47:06)



The image shows a presentation slide with a dark red header containing the text "BCJR Algorithm". Below the header, the text reads: "Step 1 : Initialize the forward and backward metrics  $\alpha_0(s)$  and  $\beta_K(s)$  using equation (16) and (17). Initial conditions for recursion: Forward recursion:  $\alpha_0(s) = \begin{cases} 1, & s = \mathbf{0} \\ 0, & s \neq \mathbf{0} \end{cases}$ ". The equation is underlined in red.

So as we said the first step is initializing alpha and betas for recursion, so since we started with all zero state, alpha at time zero for state zero is 1 and for, for other state which is state 1 it is zero.

(Refer Slide Time: 47:30)



The image shows a presentation slide with a dark red header containing the text "BCJR Algorithm". Below the header, the slide content is as follows:

Step 1 : Initialize the forward and backward metrics  $\alpha_0(s)$  and  $\beta_K(s)$  using equation (16) and (17).

Initial conditions for recursion

- Forward recursion:
$$\alpha_0(s) = \begin{cases} 1, & s = \mathbf{0} \\ 0, & s \neq \mathbf{0} \end{cases}$$
- Backward recursion:
$$\beta_K(s) = \begin{cases} 1, & s = \mathbf{0} \\ 0, & s \neq \mathbf{0} \end{cases}$$

And since we are terminate is, terminating this encoder so beta K times= 4 is 1 for state zero and it is zero for other state which is state 1, so that is the first step, initializing the forward and backward metric for time T=0 and time T end of the block in our example T=4. So once we have initialized our alphas and betas next we need to compute alphas and betas for other time instances and for that we would need.

(Refer Slide Time: 48:13)

The slide is titled "BCJR Algorithm: Example" and shows "Step 2 : Compute the branch metrics  $\gamma_l(s', s)$ ,  $l = 0, 1, \dots, K - 1$ , using equation (14)." A handwritten equation  $\gamma_l(s', s) = K_l P(u_k) e^{\frac{b_{l,k}}{T} 2(r_{l,k})}$  is written in blue ink. Below this, a list of branch metrics is shown with their numerical values:

|                      |                        |
|----------------------|------------------------|
| $\gamma_0(S_0, S_0)$ | $= e^{-0.45} = 0.6376$ |
| $\gamma_0(S_0, S_1)$ | $= e^{0.45} = 1.5683$  |
| $\gamma_1(S_0, S_0)$ | $= e^{-0.25} = 0.7788$ |
| $\gamma_1(S_0, S_1)$ | $= e^{0.25} = 1.2840$  |
| $\gamma_1(S_1, S_1)$ | $= e^{-0.75} = 0.4724$ |
| $\gamma_1(S_1, S_0)$ | $= e^{0.75} = 2.1170$  |
| $\gamma_2(S_0, S_0)$ | $= e^{0.35} = 1.4191$  |
| $\gamma_2(S_0, S_1)$ | $= e^{-0.35} = 0.7047$ |
| $\gamma_2(S_1, S_1)$ | $= e^{1.45} = 4.2631$  |
| $\gamma_2(S_1, S_0)$ | $= e^{-1.45} = 0.2346$ |
| $\gamma_3(S_0, S_0)$ | $= e^0 = 1.0$          |
| $\gamma_3(S_1, S_1)$ | $= e^{1.6} = 4.9530$   |

Our branch metric, now how do we compute our branch metric, if you recall for the AWGN channel we showed that this branch metric can be written as some constant, let us call it  $K_1$  times probability initial a-priori probability  $U_1$  and there we have explanation plus  $E_s/N_0$  2 times  $R_1 V_1$ . Now in this particular example we are assuming that a-priori is equally likely to be plus 1 or minus 1, so this probability will be half whether  $U_1$  is plus 1 or minus 1, so we can just

(Refer Slide Time: 48:58)

**BCJR Algorithm: Example**

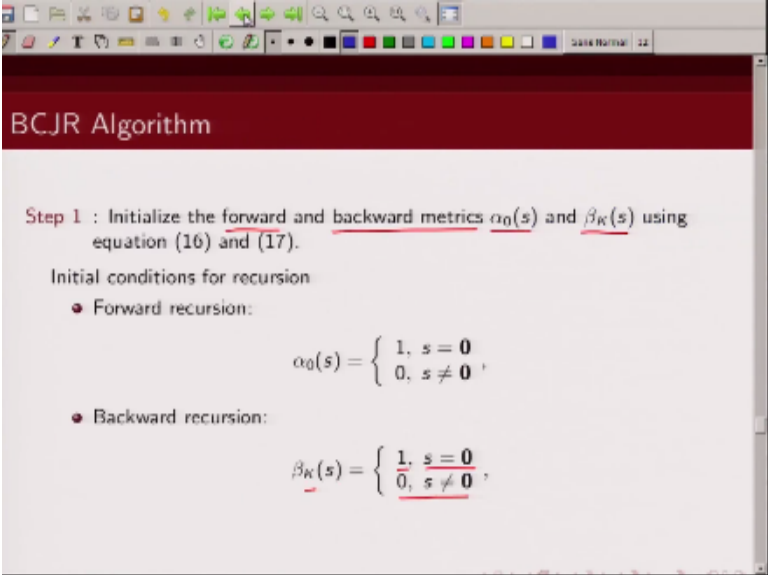
Step 2 : Compute the branch metrics  $\gamma_l(s', s)$ ,  $l = 0, 1, \dots, K-1$ , using equation (14).

Handwritten equation:  $\gamma_l(s', s) = K_l P(u_l) e^{\frac{E_s}{N_0} 2(r_l v_l)}$   
 $= K_l e^{\frac{E_s}{N_0} 2(r_l v_l)}$

|                      |                        |
|----------------------|------------------------|
| $\gamma_0(S_0, S_0)$ | $= e^{-0.45} = 0.6376$ |
| $\gamma_0(S_0, S_1)$ | $= e^{0.45} = 1.5683$  |
| $\gamma_1(S_0, S_0)$ | $= e^{-0.25} = 0.7788$ |
| $\gamma_1(S_0, S_1)$ | $= e^{0.25} = 1.2840$  |
| $\gamma_1(S_1, S_1)$ | $= e^{-0.75} = 0.4724$ |
| $\gamma_1(S_1, S_0)$ | $= e^{0.75} = 2.1170$  |
| $\gamma_2(S_0, S_0)$ | $= e^{0.35} = 1.4191$  |
| $\gamma_2(S_0, S_1)$ | $= e^{-0.35} = 0.7047$ |
| $\gamma_2(S_1, S_1)$ | $= e^{1.45} = 4.2631$  |
| $\gamma_2(S_1, S_0)$ | $= e^{-1.45} = 0.2346$ |
| $\gamma_3(S_0, S_0)$ | $= e^0 = 1.0$          |
| $\gamma_3(S_1, S_0)$ | $= e^{1.6} = 4.9530$   |

So this will be a constant, so we can just include this in a constant thing and we can just ignore this term, so what we need to compute, to compute the branch metric is basically some  $K^2$  times explanation plus  $E_s/N_0$  2 times  $R_1 \cdot V_1$  now in our example  $E_s/N_0$  is  $1/4$  if you just go back  $E_s/N_0$  is  $1/4$ .

(Refer Slide Time: 49:23)



**BCJR Algorithm**

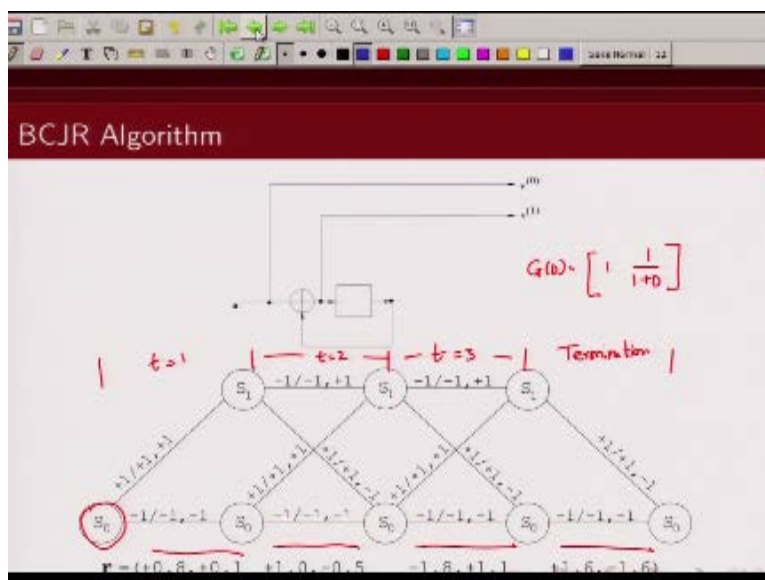
**Step 1 :** Initialize the forward and backward metrics  $\alpha_0(s)$  and  $\beta_K(s)$  using equation (16) and (17).

Initial conditions for recursion

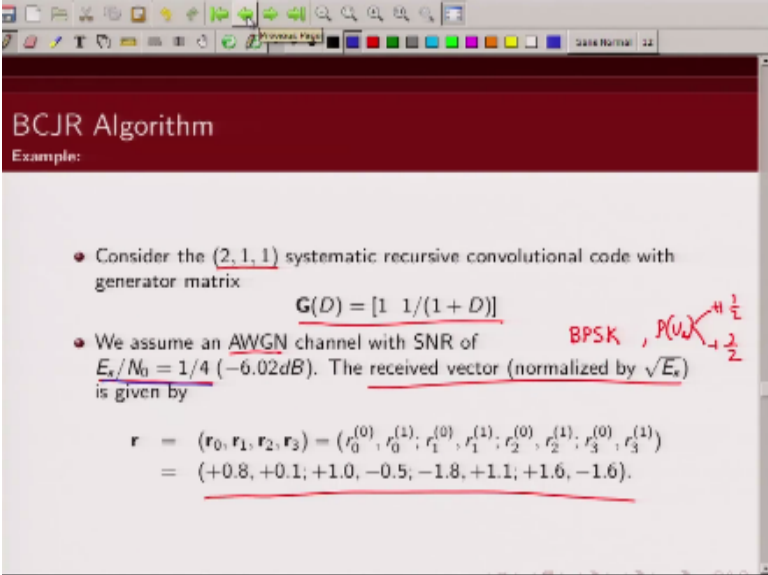
- Forward recursion:
$$\alpha_0(s) = \begin{cases} 1, & s = \mathbf{0} \\ 0, & s \neq \mathbf{0} \end{cases}$$
- Backward recursion:
$$\beta_K(s) = \begin{cases} 1, & s = \mathbf{0} \\ 0, & s \neq \mathbf{0} \end{cases}$$



(Refer Slide Time: 49:25)



(Refer Slide Time: 49:26)



The image shows a presentation slide titled "BCJR Algorithm" with the subtitle "Example:". The slide contains two bullet points and a vector equation. The first bullet point describes a (2, 1, 1) systematic recursive convolutional code with generator matrix  $G(D) = [1 \quad 1/(1+D)]$ . The second bullet point describes an AWGN channel with SNR of  $E_s/N_0 = 1/4$  (-6.02dB) and a received vector (normalized by  $\sqrt{E_s}$ ) given by  $\mathbf{r} = (r_0^{(0)}, r_0^{(1)}, r_1^{(0)}, r_1^{(1)}, r_2^{(0)}, r_2^{(1)}, r_3^{(0)}, r_3^{(1)}) = (+0.8, +0.1, +1.0, -0.5, -1.8, +1.1, +1.6, -1.6)$ . Handwritten red notes include "BPSK,  $P(u) = \begin{matrix} +1 \\ -1 \end{matrix}$ " and a red underline under the vector equation.

**BCJR Algorithm**  
Example:

- Consider the (2, 1, 1) systematic recursive convolutional code with generator matrix  $G(D) = [1 \quad 1/(1+D)]$
- We assume an AWGN channel with SNR of  $E_s/N_0 = 1/4$  (-6.02dB). The received vector (normalized by  $\sqrt{E_s}$ ) is given by

$$\mathbf{r} = (r_0^{(0)}, r_0^{(1)}, r_1^{(0)}, r_1^{(1)}, r_2^{(0)}, r_2^{(1)}, r_3^{(0)}, r_3^{(1)}) = (+0.8, +0.1, +1.0, -0.5, -1.8, +1.1, +1.6, -1.6)$$

So then

(Refer Slide Time: 49:32)

**BCJR Algorithm: Example**

Step 2 : Compute the branch metrics  $\gamma_l(s', s)$ ,  $l = 0, 1, \dots, K-1$ , using equation (14).

|                      |                        |
|----------------------|------------------------|
| $\gamma_0(S_0, S_0)$ | $= e^{-0.45} = 0.6376$ |
| $\gamma_0(S_0, S_1)$ | $= e^{0.45} = 1.5683$  |
| $\gamma_1(S_0, S_0)$ | $= e^{-0.25} = 0.7788$ |
| $\gamma_1(S_0, S_1)$ | $= e^{0.25} = 1.2840$  |
| $\gamma_1(S_1, S_1)$ | $= e^{-0.75} = 0.4724$ |
| $\gamma_1(S_1, S_0)$ | $= e^{0.75} = 2.1170$  |
| $\gamma_2(S_0, S_0)$ | $= e^{0.35} = 1.4191$  |
| $\gamma_2(S_0, S_1)$ | $= e^{-0.35} = 0.7047$ |
| $\gamma_2(S_1, S_1)$ | $= e^{1.45} = 4.2631$  |
| $\gamma_2(S_1, S_0)$ | $= e^{-1.45} = 0.2346$ |
| $\gamma_3(S_0, S_0)$ | $= e^0 = 1.0$          |
| $\gamma_3(S_1, S_0)$ | $= e^{1.6} = 4.9530$   |

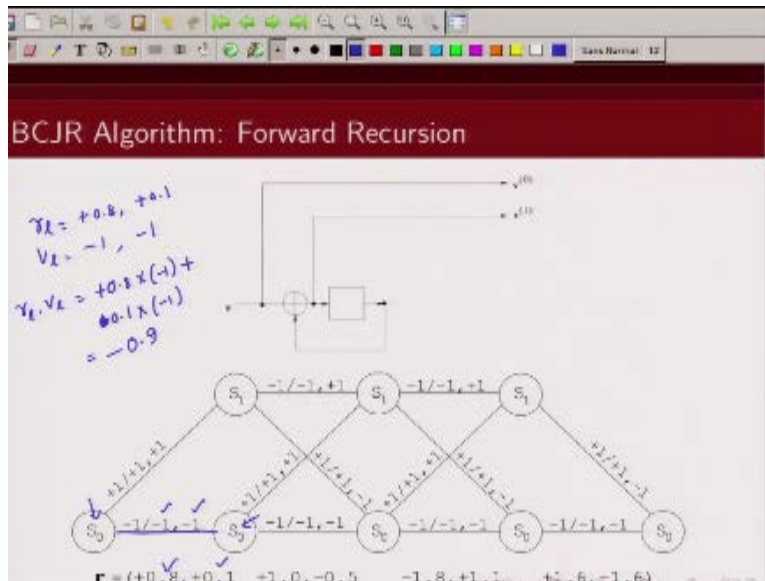
$$\gamma_l(s', s) = K_l p(U_l) e^{\frac{E_s}{N_0} 2(r_{l-1} - v_l)}$$

$$= K_l e^{\frac{E_s}{N_0} 2(r_{l-1} - v_l)}$$

$$= e^{\frac{r_{l-1} - v_l}{2}}$$

We can write this as, let us ignore the constant term and I can write this as  $R_1 \cdot V_1/2$ . Now how do we compute  $R_1 \cdot V_1$  let us take an example, let us take this gamma at time  $T=0$  when the initial state is  $S_0$  and final state is  $S_0$ , so what is this.

(Refer Slide Time: 50:00)



This corresponds to branch metric for this path, this is time  $T=0$  initial state is  $S_0$  final state is zero, now what is  $R_1$ , what is a receiver sequence corresponding to this trellis section. The receive sequence is given by this, this is plus point 8 and plus point 1, now what is the  $V_1$  corresponding to this transition, this is minus 1 and minus 1 so  $V_1$  is minus 1 and minus 1, then this product can be written as plus point 8 into minus 1 plus point, plus point 1 into minus 1 so this will be -0.9, so  $r_1 \cdot v_1$  is -0.9.

(Refer Slide Time: 51:00)

**BCJR Algorithm: Example**

Step 2 : Compute the branch metrics  $\gamma_l(s', s)$ ,  $l = 0, 1, \dots, K - 1$ , using equation (14).

$\gamma_0(S_0, S_0) = e^{-0.45} = 0.6376$

$\gamma_0(S_0, S_1) = e^{0.45} = 1.5683$

$\gamma_1(S_0, S_0) = e^{-0.25} = 0.7788$

$\gamma_1(S_0, S_1) = e^{0.25} = 1.2840$

$\gamma_1(S_1, S_1) = e^{-0.75} = 0.4724$

$\gamma_1(S_1, S_0) = e^{0.75} = 2.1170$

$\gamma_2(S_0, S_0) = e^{0.35} = 1.4191$

$\gamma_2(S_0, S_1) = e^{-0.35} = 0.7047$

$\gamma_2(S_1, S_1) = e^{1.45} = 4.2631$

$\gamma_2(S_1, S_0) = e^{-1.45} = 0.2346$

$\gamma_3(S_0, S_0) = e^0 = 1.0$

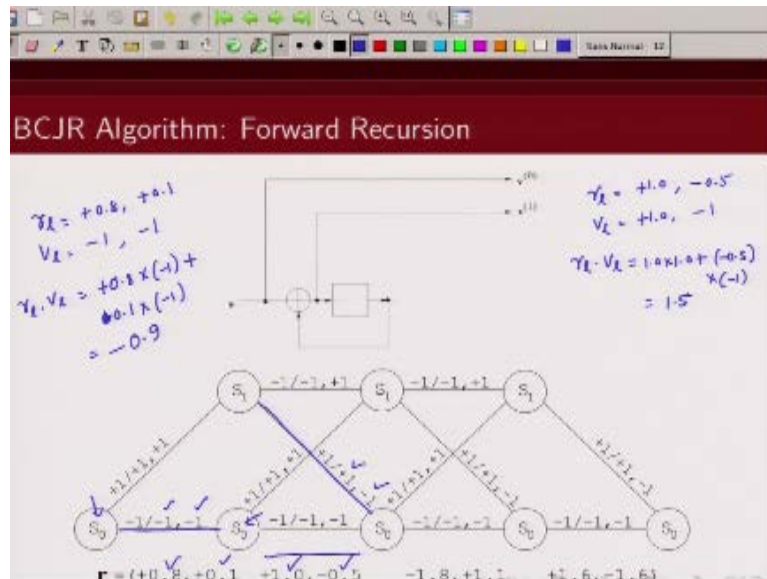
$\gamma_3(S_1, S_1) = e^{1.6} = 4.9530$

Handwritten formula:  $\gamma_l(s', s) = K_l P(u_k) e^{\frac{r_k}{2} 2(r_k - u_k)}$

Handwritten derivation:  $= K_l e^{\frac{r_k}{2} 2(r_k - u_k)} = e^{\frac{r_k - u_k}{2}}$

So if you plug that in here  $-0.9/2$ , so this  $-0.45$  which is given by this. Now you can take any other example, let us just take this example,  $\gamma_1, S_1, S_0$ , now what is this?

(Refer Slide Time: 51:20)



So  $\gamma_1$  is time instance  $t=1$  so this we are talking about this, and initial state is  $S_1$ , final state is  $S_0$ . So we are talking about this transition. Now what is  $r_1$  corresponding to this transition it is +1 and -0.5. So this is +1 and -0.5 and what is  $v_1$  corresponding to this transition, it is given by +1 and -1. So  $v_1$  is +1 and -1. So then what would be  $r_1 \cdot v_1$  in this example, it is  $1.0 \times 1.0 + (-0.5) \times (-1)$  so this will be 1.5, so you plug that in here.

(Refer Slide Time: 52:18)

**BCJR Algorithm: Example**

Step 2 : Compute the branch metrics  $\gamma_l(s', s)$ ,  $l = 0, 1, \dots, K - 1$ , using equation (14).

$\gamma_0(S_0, S_0) = e^{-0.45} = 0.6376$

$\gamma_0(S_0, S_1) = e^{0.45} = 1.5683$

$\gamma_1(S_0, S_0) = e^{-0.25} = 0.7788$

$\gamma_1(S_0, S_1) = e^{0.25} = 1.2840$

$\gamma_1(S_1, S_1) = e^{-0.75} = 0.4724$

$\gamma_1(S_1, S_0) = e^{0.75} = 2.1170$

$\gamma_2(S_0, S_0) = e^{0.35} = 1.4191$

$\gamma_2(S_0, S_1) = e^{-0.35} = 0.7047$

$\gamma_2(S_1, S_1) = e^{1.45} = 4.2631$

$\gamma_2(S_1, S_0) = e^{-1.45} = 0.2346$

$\gamma_3(S_0, S_0) = e^0 = 1.0$

$\gamma_3(S_1, S_0) = e^{1.6} = 4.9530$

$\gamma_l(s', s) = K_l P(u_k) e^{\frac{b_k}{4} 2 (r_k - v_k)}$   
 $= K_l e^{\frac{E_s}{4} 2 (r_k - v_k)}$   
 $= e^{\frac{E_s}{2} (r_k - v_k)}$

1.5/2 so this will be  $e^{0.75}$  and that is what it is okay. So I hope it is clear how we can compute the branch metric.

(Refer Slide Time: 52:33)

**BCJR Algorithm: Example**

Step 2 : Compute the branch metrics  $\gamma_l(s', s)$ ,  $l = 0, 1, \dots, K - 1$ , using equation (14).

|                      |                        |
|----------------------|------------------------|
| $\gamma_0(S_0, S_0)$ | $= e^{-0.45} = 0.6376$ |
| $\gamma_0(S_0, S_1)$ | $= e^{0.45} = 1.5683$  |
| $\gamma_1(S_0, S_0)$ | $= e^{-0.25} = 0.7788$ |
| $\gamma_1(S_0, S_1)$ | $= e^{0.25} = 1.2840$  |
| $\gamma_1(S_1, S_1)$ | $= e^{-0.75} = 0.4724$ |
| $\gamma_1(S_1, S_0)$ | $= e^{0.75} = 2.1170$  |
| $\gamma_2(S_0, S_0)$ | $= e^{0.35} = 1.4191$  |
| $\gamma_2(S_0, S_1)$ | $= e^{-0.35} = 0.7047$ |
| $\gamma_2(S_1, S_1)$ | $= e^{1.45} = 4.2631$  |
| $\gamma_2(S_1, S_0)$ | $= e^{-1.45} = 0.2346$ |
| $\gamma_3(S_0, S_0)$ | $= e^0 = 1.0$          |
| $\gamma_3(S_1, S_1)$ | $= e^{1.6} = 4.9530$   |

$$\gamma_l(s', s) = K_l P(u_k) e^{\frac{b_k}{4} 2(x_k, v_k)}$$

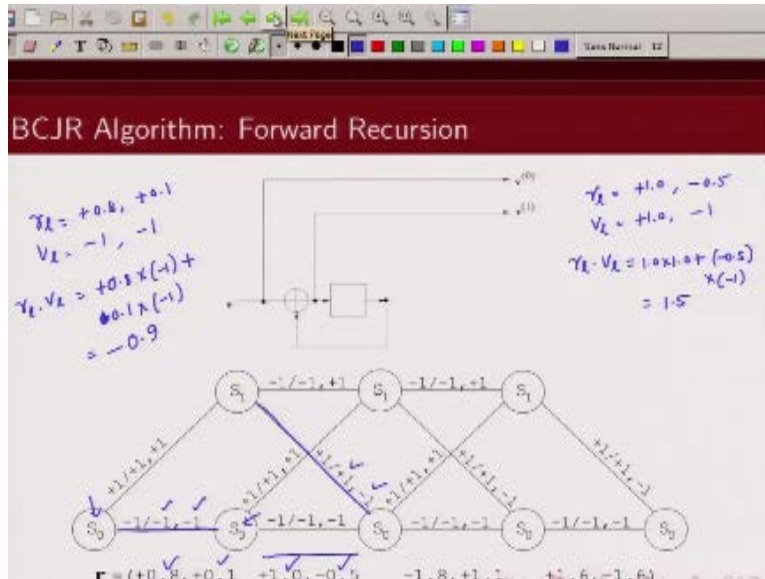
$$= K_l e^{\frac{E_k}{4} 2(x_k, v_k)}$$

$$= e^{\frac{E_k - v_k}{2}}$$

Please note we have ignored the common term which is common for computation of all of them we are just computing the term which will be different based on what state transition we are, that we are considering. So similarly we can compute gamma's for other time instance  $t=1$ ,  $t=2$  and for all other valid state transitions.

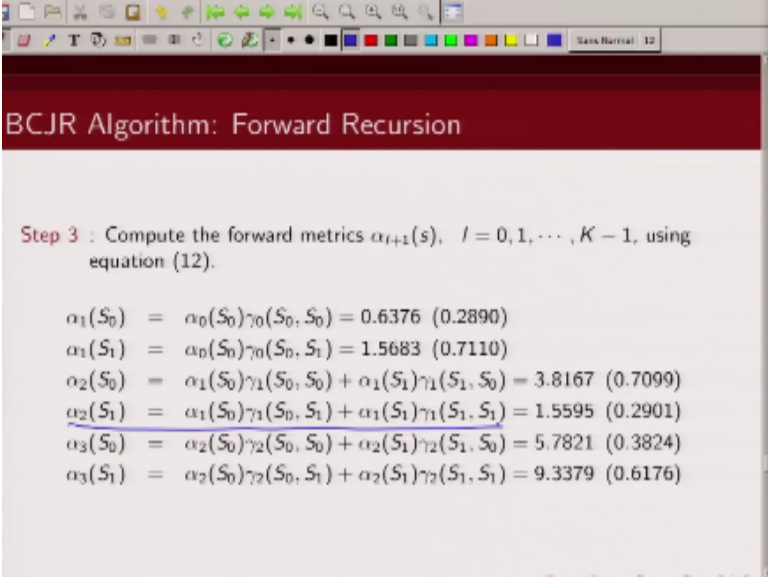


(Refer Slide Time: 52:57)



Now the next step is once we know our  $\gamma$  we have already initialized our alphas and betas so the next step would be to compute alphas and betas. And this is shown here, now we have already illustrated how we can compute our alphas and betas.

(Refer Slide Time: 53:20)



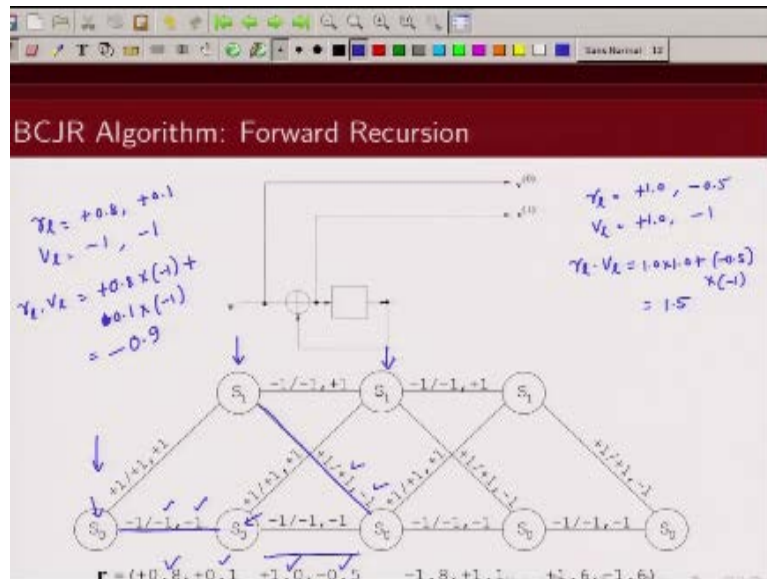
**BCJR Algorithm: Forward Recursion**

**Step 3** : Compute the forward metrics  $\alpha_{l+1}(s)$ ,  $l = 0, 1, \dots, K - 1$ , using equation (12).

$$\begin{aligned}\alpha_1(S_0) &= \alpha_0(S_0)\gamma_0(S_0, S_0) = 0.6376 \quad (0.2890) \\ \alpha_1(S_1) &= \alpha_0(S_0)\gamma_0(S_0, S_1) = 1.5683 \quad (0.7110) \\ \alpha_2(S_0) &= \alpha_1(S_0)\gamma_1(S_0, S_0) + \alpha_1(S_1)\gamma_1(S_1, S_0) = 3.8167 \quad (0.7099) \\ \alpha_2(S_1) &= \alpha_1(S_0)\gamma_1(S_0, S_1) + \alpha_1(S_1)\gamma_1(S_1, S_1) = 1.5595 \quad (0.2901) \\ \alpha_3(S_0) &= \alpha_2(S_0)\gamma_2(S_0, S_0) + \alpha_2(S_1)\gamma_2(S_1, S_0) = 5.7821 \quad (0.3824) \\ \alpha_3(S_1) &= \alpha_2(S_0)\gamma_2(S_0, S_1) + \alpha_2(S_1)\gamma_2(S_1, S_1) = 9.3379 \quad (0.6176)\end{aligned}$$

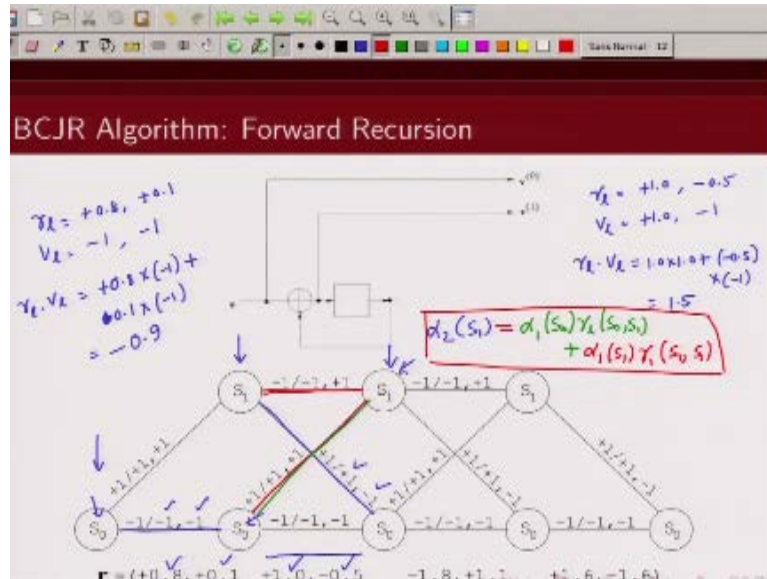
When we explained how to compute these forward recursion and backward recursion, we can take another example let us just take this example,  $\alpha_2$  at state  $S_1$ .

(Refer Slide Time: 53:36)



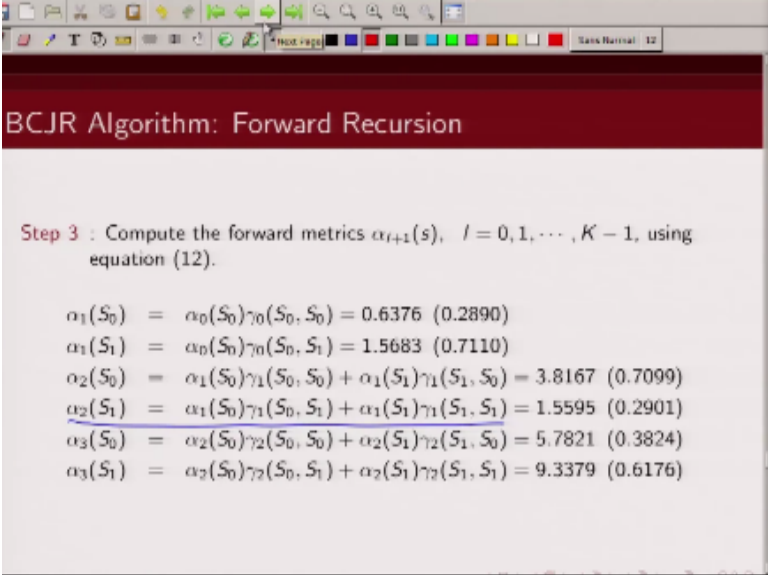
So  $\alpha_2$  will correspond to so this is  $\alpha$  at 0  $\alpha$  at 1 so this is  $\alpha$  at 2. So we are interested to calculate  $\alpha_2$ .

(Refer Slide Time: 53:47)



At state  $S_1$ , so we are interested to calculate  $\alpha_2(S_1)$ , so we are interested to calculate alpha value here. Now what are the transitions that are ending at this state, one is this, another is this, so there will be two terms in the alpha computation of this, one corresponding to this transition which is given by  $\alpha_1(S_0) \gamma_1(S_0, S_1)$  + there will be another term corresponding to this transition, this will be  $\alpha_1(S_1) \gamma_1$  initial state  $S_1$  next state  $S_1$ . So this is the value of  $\alpha_2$  at state  $S_1$ . And that is what we have got here.

(Refer Slide Time: 55:01)



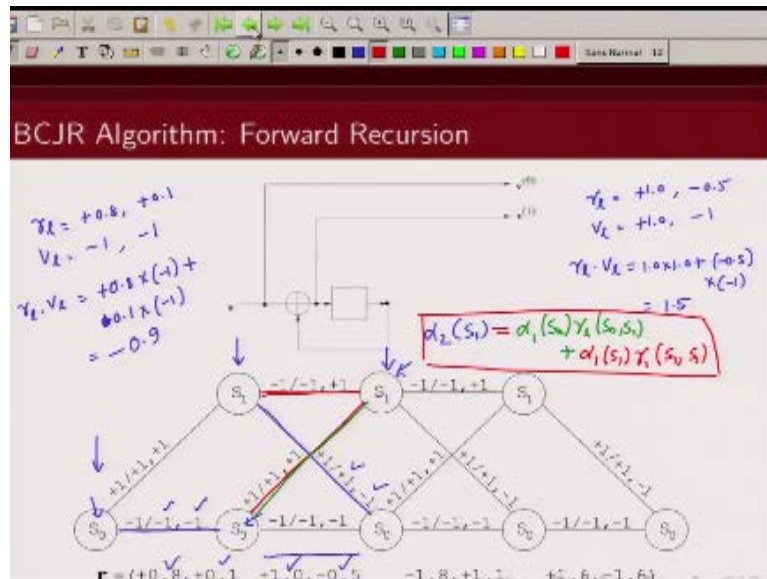
The image shows a presentation slide with a dark red header containing the text "BCJR Algorithm: Forward Recursion". Below the header, the text reads "Step 3 : Compute the forward metrics  $\alpha_{l+1}(s)$ ,  $l = 0, 1, \dots, K - 1$ , using equation (12)." followed by a list of equations. The equation for  $\alpha_2(S_1)$  is underlined in blue.

Step 3 : Compute the forward metrics  $\alpha_{l+1}(s)$ ,  $l = 0, 1, \dots, K - 1$ , using equation (12).

$$\begin{aligned}\alpha_1(S_0) &= \alpha_0(S_0)\gamma_0(S_0, S_0) = 0.6376 \quad (0.2890) \\ \alpha_1(S_1) &= \alpha_0(S_0)\gamma_0(S_0, S_1) = 1.5683 \quad (0.7110) \\ \alpha_2(S_0) &= \alpha_1(S_0)\gamma_1(S_0, S_0) + \alpha_1(S_1)\gamma_1(S_1, S_0) = 3.8167 \quad (0.7099) \\ \underline{\alpha_2(S_1)} &= \alpha_1(S_0)\gamma_1(S_0, S_1) + \alpha_1(S_1)\gamma_1(S_1, S_1) = 1.5595 \quad (0.2901) \\ \alpha_3(S_0) &= \alpha_2(S_0)\gamma_2(S_0, S_0) + \alpha_2(S_1)\gamma_2(S_1, S_0) = 5.7821 \quad (0.3824) \\ \alpha_3(S_1) &= \alpha_2(S_0)\gamma_2(S_0, S_1) + \alpha_2(S_1)\gamma_2(S_1, S_1) = 9.3379 \quad (0.6176)\end{aligned}$$

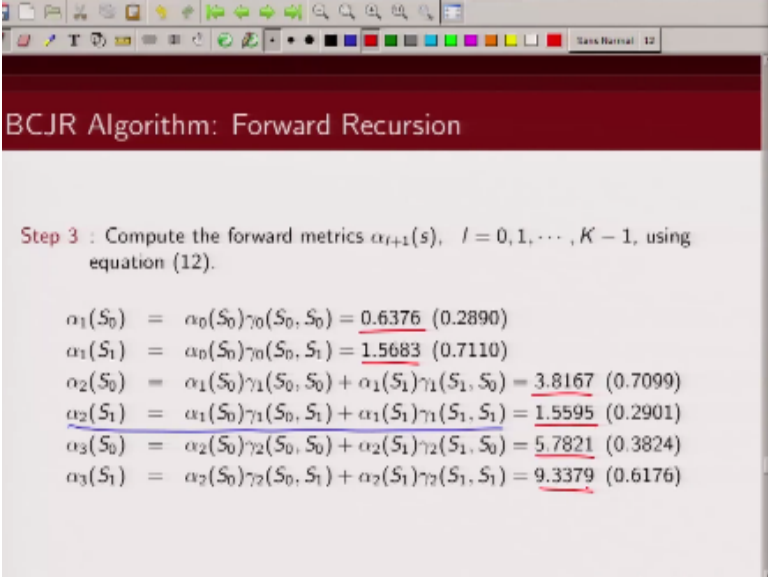
You can check  $\alpha_1(S_0) \gamma(S_0S_1)$ .

(Refer Slide Time: 55:05)



$\gamma(S_0 S_1)$  and the next term is  $\alpha(S_1) \gamma(S_1 S_1)$

(Refer Slide Time: 55:11)



**BCJR Algorithm: Forward Recursion**

**Step 3 :** Compute the forward metrics  $\alpha_{l+1}(s)$ ,  $l = 0, 1, \dots, K-1$ , using equation (12).

$$\begin{aligned}\alpha_1(S_0) &= \alpha_0(S_0)\gamma_0(S_0, S_0) = \underline{0.6376} \quad (0.2890) \\ \alpha_1(S_1) &= \alpha_0(S_0)\gamma_0(S_0, S_1) = \underline{1.5683} \quad (0.7110) \\ \alpha_2(S_0) &= \alpha_1(S_0)\gamma_1(S_0, S_0) + \alpha_1(S_1)\gamma_1(S_1, S_0) = \underline{3.8167} \quad (0.7099) \\ \alpha_2(S_1) &= \alpha_1(S_0)\gamma_1(S_0, S_1) + \alpha_1(S_1)\gamma_1(S_1, S_1) = \underline{1.5595} \quad (0.2901) \\ \alpha_3(S_0) &= \alpha_2(S_0)\gamma_2(S_0, S_0) + \alpha_2(S_1)\gamma_2(S_1, S_0) = \underline{5.7821} \quad (0.3824) \\ \alpha_3(S_1) &= \alpha_2(S_0)\gamma_2(S_0, S_1) + \alpha_2(S_1)\gamma_2(S_1, S_1) = \underline{9.3379} \quad (0.6176)\end{aligned}$$

$\alpha(S_1) \gamma(S_1 S_1)$ . So like this we can compute the values of alphas and these values that you see are basically the values of alpha computing the, computed this way. Now we can also normalize the values of alphas because alphas are some of the alphas were all state should add up to 1.

(Refer Slide Time: 55:35)

BCJR Algorithm: Forward Recursion

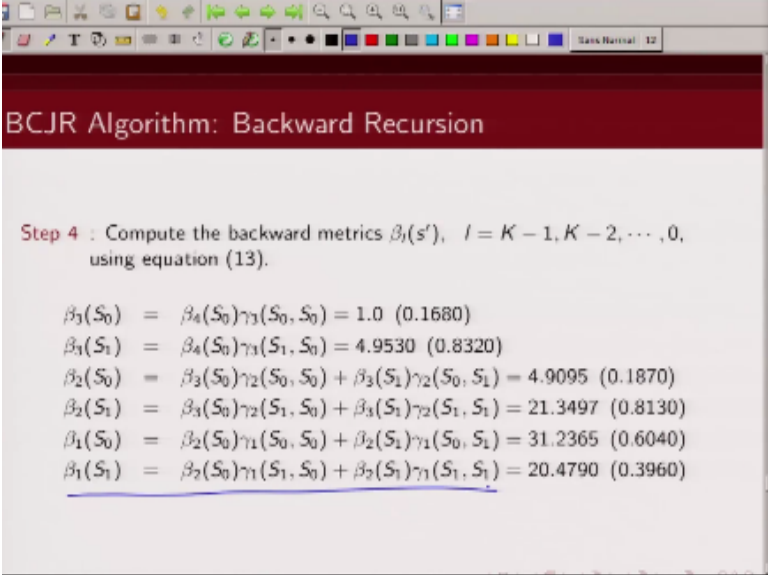
Step 3 : Compute the forward metrics  $\alpha_{l+1}(s)$ ,  $l = 0, 1, \dots, K-1$ , using equation (12).

$$\begin{aligned} \alpha_1(S_0) &= \alpha_0(S_0)\gamma_0(S_0, S_0) = \underline{0.6376} \quad (0.2890) \leftarrow \frac{0.6376}{0.6376+1.5683} \\ \alpha_1(S_1) &= \alpha_0(S_0)\gamma_0(S_0, S_1) = \underline{1.5683} \quad (0.7110) \leftarrow \frac{1.5683}{0.6376+1.5683} \\ \alpha_2(S_0) &= \alpha_1(S_0)\gamma_1(S_0, S_0) + \alpha_1(S_1)\gamma_1(S_1, S_0) = \underline{3.8167} \quad (0.7099) \\ \alpha_2(S_1) &= \alpha_1(S_0)\gamma_1(S_0, S_1) + \alpha_1(S_1)\gamma_1(S_1, S_1) = \underline{1.5595} \quad (0.2901) \\ \alpha_3(S_0) &= \alpha_2(S_0)\gamma_2(S_0, S_0) + \alpha_2(S_1)\gamma_2(S_1, S_0) = \underline{5.7821} \quad (0.3824) \\ \alpha_3(S_1) &= \alpha_2(S_0)\gamma_2(S_0, S_1) + \alpha_2(S_1)\gamma_2(S_1, S_1) = \underline{9.3379} \quad (0.6176) \end{aligned}$$

So in the bracket that you see here these are the normalized values of alpha. So how do I get this, so this is  $0.6376 / (0.6376 + 1.5683)$  so this is this quantity. Similarly this is  $1.5683 / (0.6376 + 1.5683)$  so this is this quantity. So these are the actual values of  $\alpha$  and these are I normalized values because from, these are sum of probability should add up to 1 so I can, I can take this value so when I compute alphas or I can just work with these values or I can work with these values, it is just a scaled version so does not make a difference. Expect for implementation purpose you may want to scale them, add them up to 1 so that the values do not blow up.



(Refer Slide Time: 56:36)



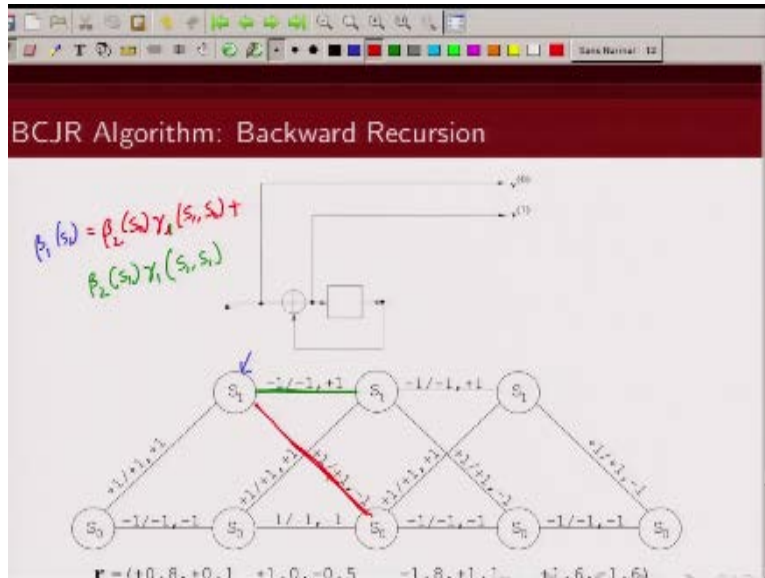
**BCJR Algorithm: Backward Recursion**

**Step 4** : Compute the backward metrics  $\beta_l(s^l)$ ,  $l = K - 1, K - 2, \dots, 0$ , using equation (13).

$$\begin{aligned}\beta_3(S_0) &= \beta_4(S_0)\gamma_3(S_0, S_0) = 1.0 \quad (0.1680) \\ \beta_3(S_1) &= \beta_4(S_0)\gamma_3(S_1, S_0) = 4.9530 \quad (0.8320) \\ \beta_2(S_0) &= \beta_3(S_0)\gamma_2(S_0, S_0) + \beta_3(S_1)\gamma_2(S_0, S_1) = 4.9095 \quad (0.1870) \\ \beta_2(S_1) &= \beta_3(S_0)\gamma_2(S_1, S_0) + \beta_3(S_1)\gamma_2(S_1, S_1) = 21.3497 \quad (0.8130) \\ \beta_1(S_0) &= \beta_2(S_0)\gamma_1(S_0, S_0) + \beta_2(S_1)\gamma_1(S_0, S_1) = 31.2365 \quad (0.6040) \\ \beta_1(S_1) &= \beta_2(S_0)\gamma_1(S_1, S_0) + \beta_2(S_1)\gamma_1(S_1, S_1) = 20.4790 \quad (0.3960)\end{aligned}$$

So once we have computed alphas we can follow the same procedure to compute  $\beta$ , so  $\beta$  computation is given here again we can as an example we can take one particular case, let us just consider this  $\beta_1$  at state  $S_1$ .

(Refer Slide Time: 56:54)



So this is  $\beta_0, \beta_1$  at state  $S_1$  so we are interested in computing  $\beta_1(S_1)$ . Now what are the transitions that are ending in state  $S_1$ , one of them is this, the other one is this. So what is the contribution from this, this transition this can be written as  $\beta_2(S_0)$  times  $\gamma_1(S_1, S_0)$  + and the contribution from this will be  $\beta_2(S_1)$  times  $\gamma$  of this, this is  $\gamma_1$  this is 1,  $\gamma_1(S_1, S_1)$ . So this, the one in green is corresponding to this transition, the one in red is due to this transition okay this transition. So  $\beta(S_1)$  can be written as  $\beta_2(S_0)\gamma_1(S_1, S_0) + \beta_2(S_1)\gamma_1(S_1, S_1)$ .

(Refer Slide Time: 58:15)

**BCJR Algorithm: Backward Recursion**

**Step 4 :** Compute the backward metrics  $\beta_l(s^l)$ ,  $l = K - 1, K - 2, \dots, 0$ , using equation (13).

✓  $\beta_3(S_0) = \beta_4(S_0) \gamma_3(S_0, S_0) = 1.0 \text{ (0.1680)}$

✓  $\beta_3(S_1) = \beta_4(S_0) \gamma_3(S_1, S_0) = 4.9530 \text{ (0.8320)}$

✓  $\beta_2(S_0) = \beta_3(S_0) \gamma_2(S_0, S_0) + \beta_3(S_1) \gamma_2(S_0, S_1) = 4.9095 \text{ (0.1870)}$

✓  $\beta_2(S_1) = \beta_3(S_0) \gamma_2(S_1, S_0) + \beta_3(S_1) \gamma_2(S_1, S_1) = 21.3497 \text{ (0.8130)}$

✓  $\beta_1(S_0) = \beta_2(S_0) \gamma_1(S_0, S_0) + \beta_2(S_1) \gamma_1(S_0, S_1) = 31.2365 \text{ (0.6040)}$

✓  $\beta_1(S_1) = \beta_2(S_0) \gamma_1(S_1, S_0) + \beta_2(S_1) \gamma_1(S_1, S_1) = 20.4790 \text{ (0.3960)}$

And that is what we have  $\beta_2(S_0) \gamma_1(S_1, S_0)$ ,  $\beta_2(S_1) \gamma_1(S_1, S_1)$ . We already know the values of gammas that we have computed earlier and we know the values of  $\beta$  at the end of block. What is this value, this is one because the encoder is terminated and for all other state basically  $\beta_4(S_1, S_0)$ . So this is 1 we know this values so we can compute what is  $\beta_3$  once we know the value of  $\beta_3$  then we can compute values of  $\beta_2$  because  $\beta_3$  values we will require here. Once we know the value for  $\beta_2$  we can compute the value of  $\beta_1$  they are required here. So like that basically we can recursively compute the values of betas. So now what do we have, we have the values of alphas, we have the values of beats, and we have the branch metrics gammas, next step we need to compute the APP value and what is the APP value?

(Refer Slide Time: 59:21)

**BCJR Algorithm: APP values**

$\sum_{s'} \alpha_k(s') \gamma_k(s', s) \beta_{k+1}(s)$

**Step 5 :** Compute the APP L-values  $L(u_i)$ , using equations (6) and (11).

$$L(u_0) = \ln \left\{ \frac{\alpha_0(S_0) \gamma_0(S_0, S_1) \beta_1(S_1)}{\alpha_0(S_0) \gamma_0(S_0, S_0) \beta_1(S_0)} \right\} = 0.4778$$

$$L(u_1) = \ln \left\{ \frac{\alpha_1(S_0) \gamma_1(S_0, S_1) \beta_2(S_1) + \alpha_1(S_1) \gamma_1(S_1, S_0) \beta_2(S_0)}{\alpha_1(S_0) \gamma_1(S_0, S_0) \beta_2(S_0) + \alpha_1(S_1) \gamma_1(S_1, S_1) \beta_2(S_1)} \right\} = 0.6154$$

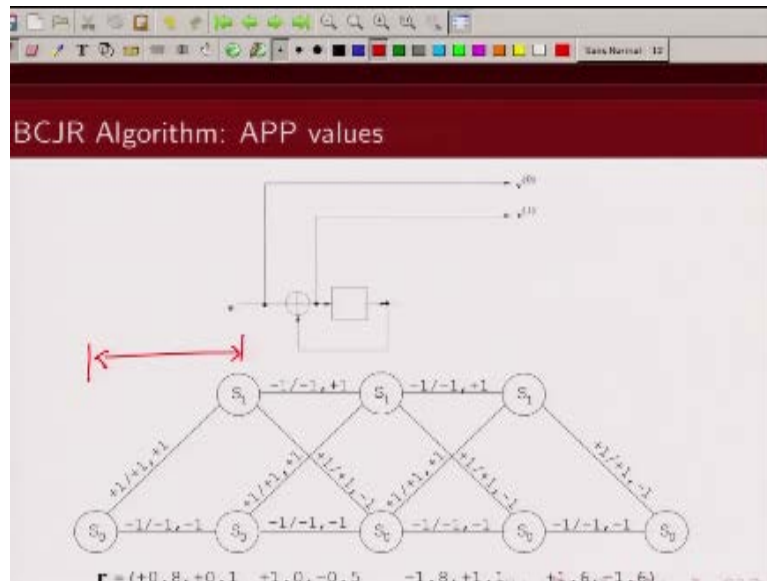
$$L(u_2) = \ln \left\{ \frac{\alpha_2(S_0) \gamma_2(S_0, S_1) \beta_3(S_1) + \alpha_2(S_1) \gamma_2(S_1, S_0) \beta_3(S_0)}{\alpha_2(S_0) \gamma_2(S_0, S_0) \beta_3(S_0) + \alpha_2(S_1) \gamma_2(S_1, S_1) \beta_3(S_1)} \right\} = -1.0301$$

**Step 6 :** Compute the hard decisions  $\hat{u}_i$  using equation (2).

$$\hat{u} = (+1, +1, -1)$$

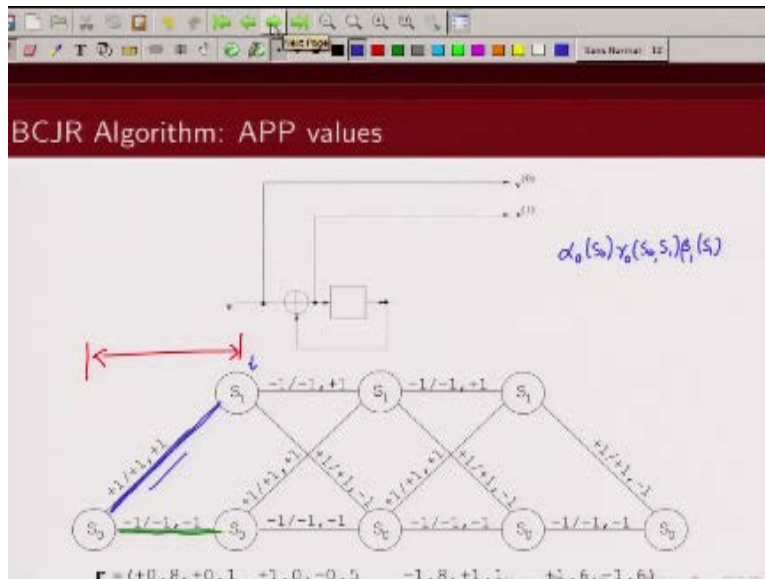
If you recall APP value  $v$  it was product of three terms alpha, gamma and betas, so product of this term. And what was the term in the numerator, we were summing over all those transitions which belongs to  $u^{b+1}$ . And in the denominator we were summing over all those transitions belonging to  $u^b$ . So let us look at how we can compute this so let us look at first case. A time, so  $u_0$  is the information sequence that we are trying to estimate for time first time instance  $t=0$ .

(Refer Slide Time: 01:00:21)



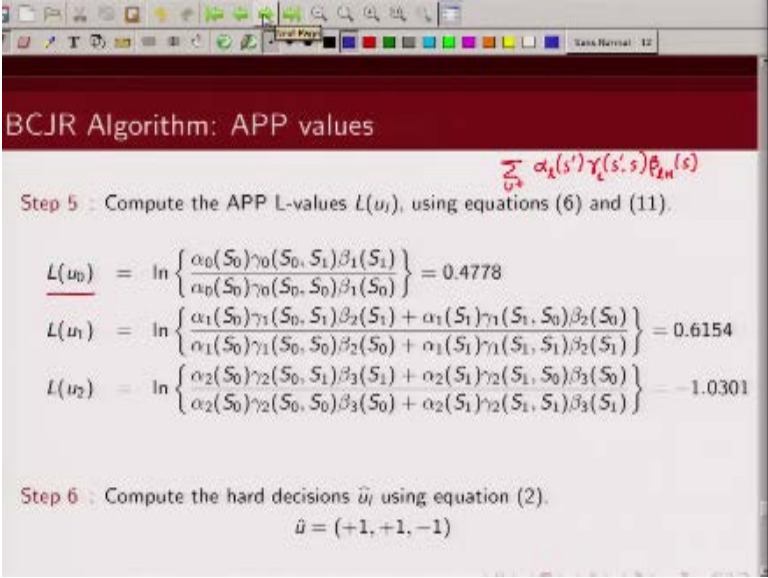
So this is, we are looking at this trellis section. Now which are the transitions corresponding to  $u_1$  being +1? So I am denoting by blue.

(Refer Slide Time: 01:00:36)



The transitions which is corresponding to  $u_1$  being +1. And I am denoting by green the transition corresponding to  $u_1$  being -1 okay. Then in the numerator then I will have one term corresponding to this transition and what would be that term, it will be  $\alpha_0(S_0)$  and  $\gamma$  corresponding to this transition which is  $\gamma_0(S_0, S_1)$  times  $\beta_1(S_1)$  so  $\alpha_0(S_0)$   $\alpha$  at this state,  $\gamma$  corresponding to this transition which is  $\gamma_0(S_0, S_1)$  and  $\beta$  corresponding to this state. So if you go back.

(Refer Slide Time: 01:01:34)



**BCJR Algorithm: APP values**

$\sum_{s'} \alpha_k(s') \gamma(s', s) \beta_{kn}(s)$

**Step 5 :** Compute the APP L-values  $L(u_i)$ , using equations (6) and (11).

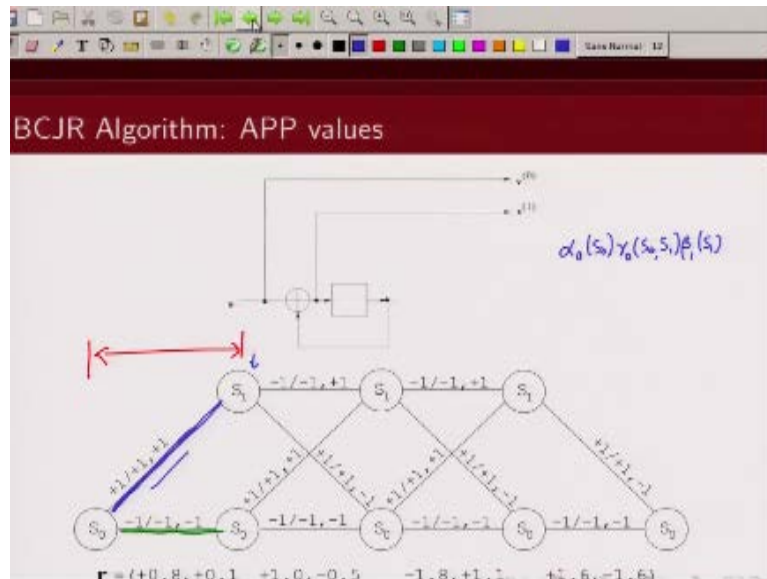
$$L(u_0) = \ln \left\{ \frac{\alpha_0(S_0) \gamma_0(S_0, S_1) \beta_1(S_1)}{\alpha_0(S_0) \gamma_0(S_0, S_0) \beta_1(S_0)} \right\} = 0.4778$$
$$L(u_1) = \ln \left\{ \frac{\alpha_1(S_0) \gamma_1(S_0, S_1) \beta_2(S_1) + \alpha_1(S_1) \gamma_1(S_1, S_0) \beta_2(S_0)}{\alpha_1(S_0) \gamma_1(S_0, S_0) \beta_2(S_0) + \alpha_1(S_1) \gamma_1(S_1, S_1) \beta_2(S_1)} \right\} = 0.6154$$
$$L(u_2) = \ln \left\{ \frac{\alpha_2(S_0) \gamma_2(S_0, S_1) \beta_3(S_1) + \alpha_2(S_1) \gamma_2(S_1, S_0) \beta_3(S_0)}{\alpha_2(S_0) \gamma_2(S_0, S_0) \beta_3(S_0) + \alpha_2(S_1) \gamma_2(S_1, S_1) \beta_3(S_1)} \right\} = -1.0301$$

**Step 6 :** Compute the hard decisions  $\hat{u}_i$  using equation (2).

$$\hat{u} = (+1, +1, -1)$$

This is what I have  $\alpha_0(S_0)$   $\gamma_0(S_0, S_1)$  and  $\beta_1(S_1)$ .

(Refer Slide Time: 01:01:43)



At this time instance is there any other transition corresponding to  $u_{i+1}$  no, there is only one transition corresponding to  $u_{i+1}$ .



(Refer Slide Time: 01:01:56)

**BCJR Algorithm: APP values**

$\sum_{u^+} \alpha_x(s') \gamma_\ell(s', s) \beta_{xH}(s)$

**Step 5 :** Compute the APP L-values  $L(u_i)$ , using equations (6) and (11).

$$L(u_0) = \ln \left\{ \frac{\alpha_0(S_0) \gamma_0(S_0, S_1) \beta_1(S_1)}{\alpha_0(S_0) \gamma_0(S_0, S_0) \beta_1(S_0)} \right\} = 0.4778$$

$u^-$

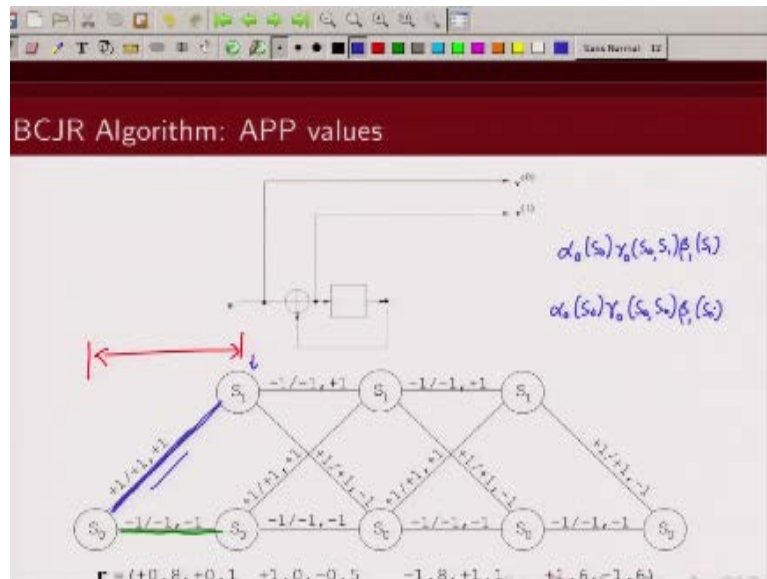
$$L(u_1) = \ln \left\{ \frac{\alpha_1(S_0) \gamma_1(S_0, S_1) \beta_2(S_1) + \alpha_1(S_1) \gamma_1(S_1, S_0) \beta_2(S_0)}{\alpha_1(S_0) \gamma_1(S_0, S_0) \beta_2(S_0) + \alpha_1(S_1) \gamma_1(S_1, S_1) \beta_2(S_1)} \right\} = 0.6154$$
$$L(u_2) = \ln \left\{ \frac{\alpha_2(S_0) \gamma_2(S_0, S_1) \beta_3(S_1) + \alpha_2(S_1) \gamma_2(S_1, S_0) \beta_3(S_0)}{\alpha_2(S_0) \gamma_2(S_0, S_0) \beta_3(S_0) + \alpha_2(S_1) \gamma_2(S_1, S_1) \beta_3(S_1)} \right\} = -1.0301$$

**Step 6 :** Compute the hard decisions  $\hat{u}_i$  using equation (2).

$$\hat{u} = (+1, +1, -1)$$

So we will now look at the denominator term, so we have to look for all those transitions corresponding  $u$  being -1.

(Refer Slide Time: 01:02:05)



And there is only one such transitions, so the denominator term would be  $\alpha_0(S_0) \gamma_0(S_0, S_0)$  and  $\beta_1(S_0)$ .

(Refer Slide Time: 01:02:22)

$\sum_{u^+} \alpha_k(s') \gamma_\ell(s', s) \beta_{kM}(s)$

**Step 5 :** Compute the APP L-values  $L(u_i)$ , using equations (6) and (11).

$$L(u_0) = \ln \left\{ \frac{\alpha_0(S_0) \gamma_0(S_0, S_1) \beta_1(S_1)}{\alpha_0(S_0) \gamma_0(S_0, S_0) \beta_1(S_0)} \right\} = 0.4778 \quad u^-$$

$$L(u_1) = \ln \left\{ \frac{\alpha_1(S_0) \gamma_1(S_0, S_1) \beta_2(S_1) + \alpha_1(S_1) \gamma_1(S_1, S_0) \beta_2(S_0)}{\alpha_1(S_0) \gamma_1(S_0, S_0) \beta_2(S_0) + \alpha_1(S_1) \gamma_1(S_1, S_1) \beta_2(S_1)} \right\} = 0.6154$$

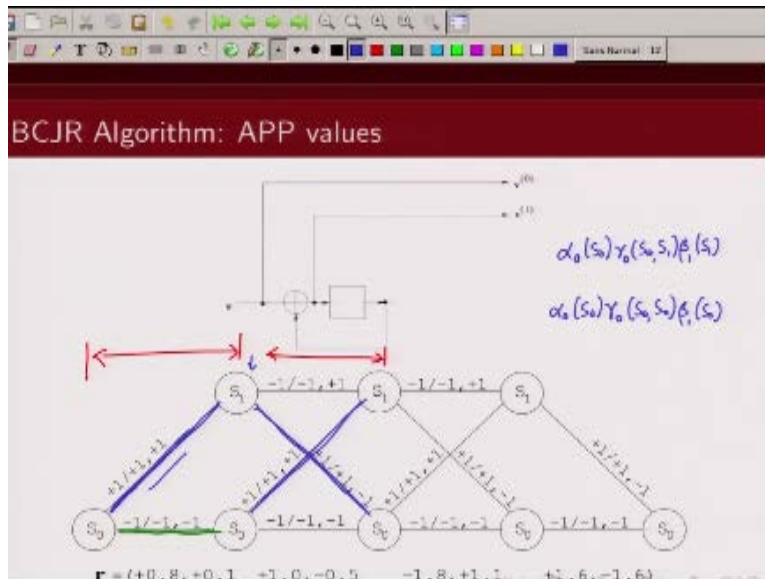
$$L(u_2) = \ln \left\{ \frac{\alpha_2(S_0) \gamma_2(S_0, S_1) \beta_3(S_1) + \alpha_2(S_1) \gamma_2(S_1, S_0) \beta_3(S_0)}{\alpha_2(S_0) \gamma_2(S_0, S_0) \beta_3(S_0) + \alpha_2(S_1) \gamma_2(S_1, S_1) \beta_3(S_1)} \right\} = -1.0301$$

**Step 6 :** Compute the hard decisions  $\hat{u}_i$  using equation (2).

$$\hat{u} = (+1, +1, -1)$$

So this is what we have  $\alpha_0(S_0)$   $\gamma_0(S_0, S_0)$  and  $\beta_1(S_0)$ . So we can then calculate the, what is the A positive to read L value. Now let us take another example, let us take for the second time instance. For second time instance we are interested in estimating what was our information sequence, information bit.

(Refer Slide Time: 01:02:54)



So we are now looking at this time instance, this time instance okay. Now what are the transitions corresponding to  $u_1$  information sequence being +1, one of them is this, you can see the information sequence is +1 that is when you go from  $S_0$  to  $S_1$  and another transition is this one. So these are the two transitions corresponding to  $u_1$  being +1. So in the numerator you will have two terms, one corresponding to  $\alpha_1(S_0) \gamma_1(S_0, S_1)$  times  $\beta_2(S_1)$  and another term corresponding to this transition which is  $\alpha_1(S_1)$  times  $\gamma_1(S_1, S_0)$  multiplied by  $\beta_2(S_0)$ .

(Refer Slide Time: 01:04:00)

**BCJR Algorithm: APP values**

$\sum_{s'} \alpha_i(s') \gamma_i(s', s) \beta_{i+1}(s)$

**Step 5 :** Compute the APP L-values  $L(u_i)$ , using equations (6) and (11).

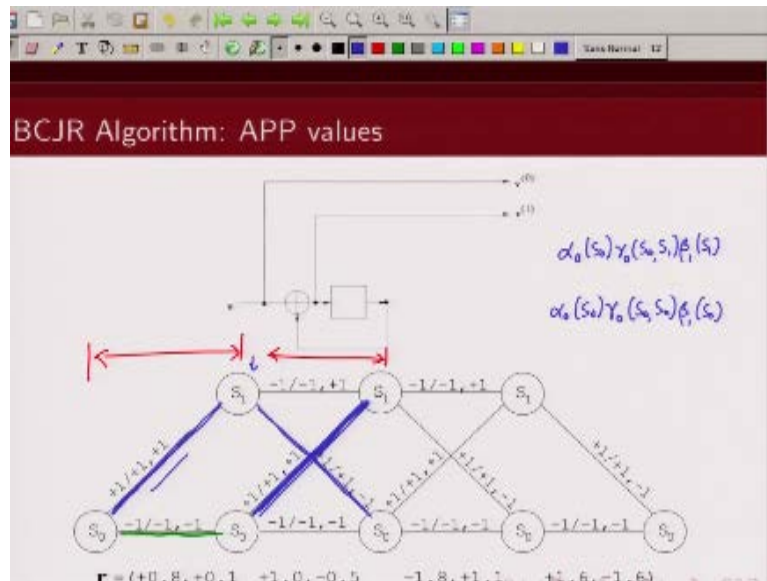
$$L(u_0) = \ln \left\{ \frac{\alpha_0(S_0) \gamma_0(S_0, S_1) \beta_1(S_1)}{\alpha_0(S_0) \gamma_0(S_0, S_0) \beta_1(S_0)} \right\} = 0.4778 \quad u^-$$
$$L(u_1) = \ln \left\{ \frac{\alpha_1(S_0) \gamma_1(S_0, S_1) \beta_2(S_1) + \alpha_1(S_1) \gamma_1(S_1, S_0) \beta_2(S_0)}{\alpha_1(S_0) \gamma_1(S_0, S_0) \beta_2(S_0) + \alpha_1(S_1) \gamma_1(S_1, S_1) \beta_2(S_1)} \right\} = 0.6154$$
$$L(u_2) = \ln \left\{ \frac{\alpha_2(S_0) \gamma_2(S_0, S_1) \beta_3(S_1) + \alpha_2(S_1) \gamma_2(S_1, S_0) \beta_3(S_0)}{\alpha_2(S_0) \gamma_2(S_0, S_0) \beta_3(S_0) + \alpha_2(S_1) \gamma_2(S_1, S_1) \beta_3(S_1)} \right\} = -1.0301$$

**Step 6 :** Compute the hard decisions  $\hat{u}_i$  using equation (2).

$$\hat{u} = (+1, +1, -1)$$

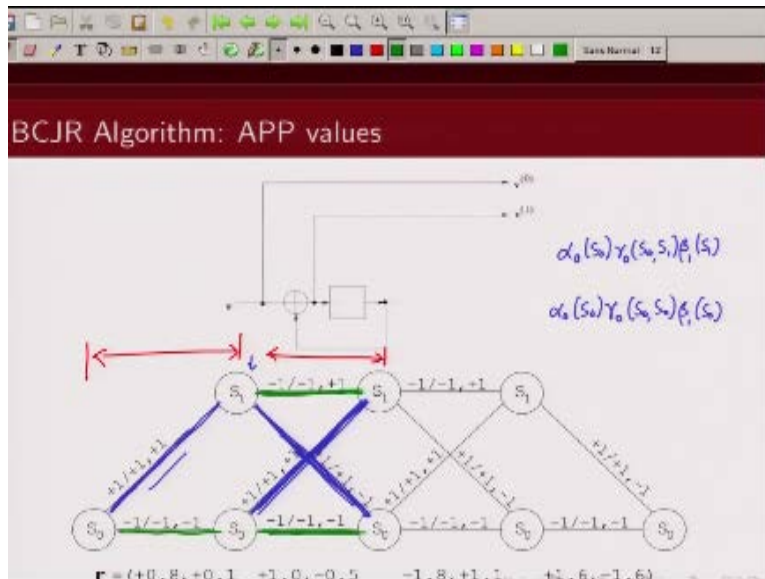
And that is what you see here, there are two terms one is  $\alpha_1(S_0) \gamma_1(S_0, S_1) \beta_2(S_1)$ .

(Refer Slide Time: 01:04:09)



This corresponds to this transition.

(Refer Slide Time: 01:04:15)



And the next term that you see here, this one corresponds to this transition okay. Now similarly in the denominator you need to look at what are the valid transitions corresponding to  $u_{i-1}$ , and what are those, one of them is this and the second one is this. So now in the numerator, denominator also you will have two terms one corresponding to this transition, other corresponding to this transition. This transition will give you  $\alpha_1(S_0) \gamma_1(S_0, S_0)$  times  $\beta_2(S_0) + \alpha_1(S_1) \gamma_1(S_1, S_1)$  times  $\beta_2(S_1)$ .

(Refer Slide Time: 01:05:13)

**BCJR Algorithm: APP values**

$\sum_{\hat{u}_i} \alpha_x(s') \gamma_\ell(s', s) \beta_{2^n}(s)$

**Step 5 :** Compute the APP L-values  $L(u_i)$ , using equations (6) and (11).

$$L(u_0) = \ln \left\{ \frac{\alpha_0(S_0) \gamma_0(S_0, S_1) \beta_1(S_1)}{\alpha_0(S_0) \gamma_0(S_0, S_0) \beta_1(S_0)} \right\} = 0.4778$$
$$L(u_1) = \ln \left\{ \frac{\alpha_1(S_0) \gamma_1(S_0, S_1) \beta_2(S_1) + \alpha_1(S_1) \gamma_1(S_1, S_0) \beta_2(S_0)}{\alpha_1(S_0) \gamma_1(S_0, S_0) \beta_2(S_0) + \alpha_1(S_1) \gamma_1(S_1, S_1) \beta_2(S_1)} \right\} = 0.6154$$
$$L(u_2) = \ln \left\{ \frac{\alpha_2(S_0) \gamma_2(S_0, S_1) \beta_3(S_1) + \alpha_2(S_1) \gamma_2(S_1, S_0) \beta_3(S_0)}{\alpha_2(S_0) \gamma_2(S_0, S_0) \beta_3(S_0) + \alpha_2(S_1) \gamma_2(S_1, S_1) \beta_3(S_1)} \right\} = -1.0301$$

**Step 6 :** Compute the hard decisions  $\hat{u}_i$  using equation (2).

$$\hat{u} = (+1, +1, -1)$$

And that is what you have here. So likewise we compute log like u ratios, APP values for all the three information bits. Now what is the final step, once we have computed the log like u ratio we will see whether these log like u ratios are greater than 0 or less than 0? If they are greater than equal to 0 we decide in favor of  $u_i+1$  otherwise we decide in favor of  $u_i-1$ . So this is 0.4778 which is greater than 0 so we decide in favor of +1. This is greater than 0 so we decide in favor of 1 and this one is less than 0 so we decide in favor of -1. So then the final decoded bits are +1, +1 and -1. So with this I will conclude this lecture. Thank you.

### **Acknowledgement**

**Ministry of Human Resource & Development**

**Prof. Satyaki Roy**

**Co-ordinator, NPTEL IIT Kanpur**

**NPTEL Team**

**Sanjay Pal**



**Ashish Singh**  
**Badal Pradhan**  
**Tapobrata Das**  
**Ram Chandra**  
**Dilip Tripathi**  
**Manoj Shrivastava**  
**Padam Shukla**  
**Sanjay Mishra**  
**Shubham Rawat**  
**Shikha Gupta**  
**K. K. Mishra**  
**Aradhana Singh**  
**Sweta**  
**Ashutosh Gairola**  
**Dilip Katiyar**  
**Sharwan**  
**Hari Ram**  
**Bhadra Rao**  
**Puneet Kumar Bajpai**  
**Lalty Dutta**  
**Ajay Kanaujia**  
**Shivendra Kumar Tiwari**

**an IIT Kanpur Production**

ssssss©copyright reserved