

An Introduction to Coding Theory
Professor Adrish Banerji
Department of Electrical Engineering
Indian Institute of Technology, Kanpur
Module 05
Lecture Number 20
Decoding of convolutional codes-II BCJR algorithm

(Refer Slide Time 00:14)

Lecture #11A: Decoding of convolutional codes-II: BCJR algorithm



We are going to continue our discussion on decoding of convolutional codes. In the last class

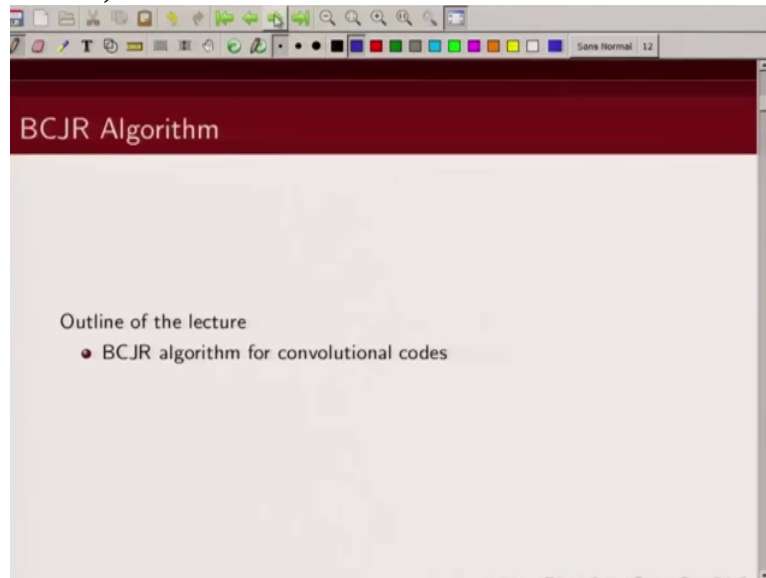
(Refer Slide Time 00:20)



we talked about Viterbi decoding. And if you recall Viterbi decoding is an efficient algorithm to compute a path to the Trellis of a convolutional code. Now it essentially finds out, Viterbi algorithm essentially finds out an estimate of the codeword because any path through the Trellis of a convolutional code is basically a codeword. Now that not necessarily minimizes

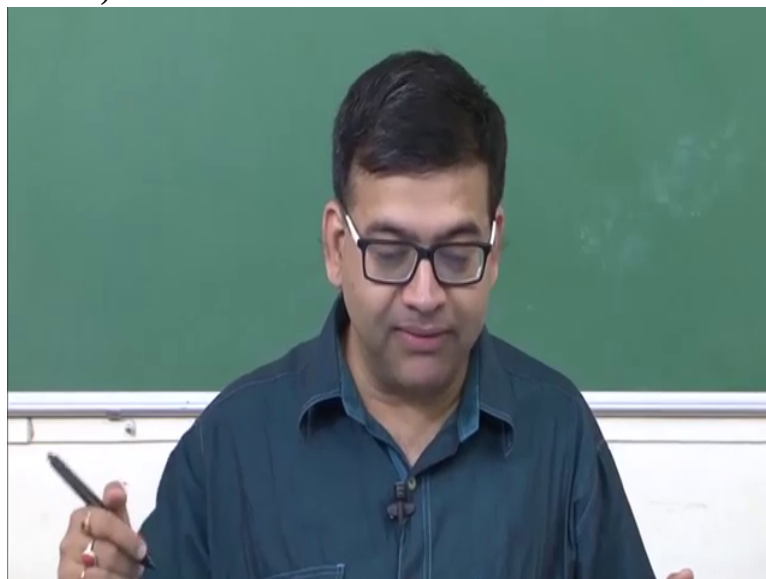
the bit error rate probability. In many applications we are interested to minimize the bit error rate. So

(Refer Slide Time 01:01)



today we are going to talk about a decoding algorithm which is basically going to minimize

(Refer Slide Time 01:09)



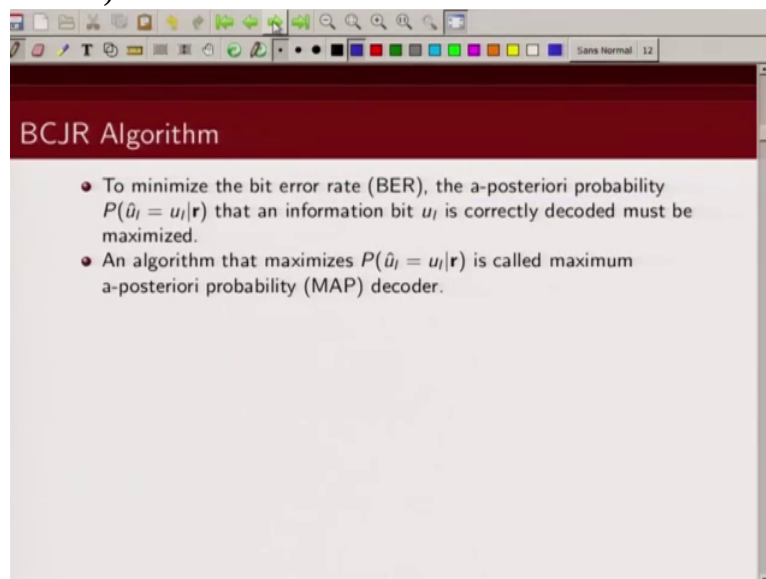
bit error rate probability, symbol error rate probability.

(Refer Slide Time 01:12)



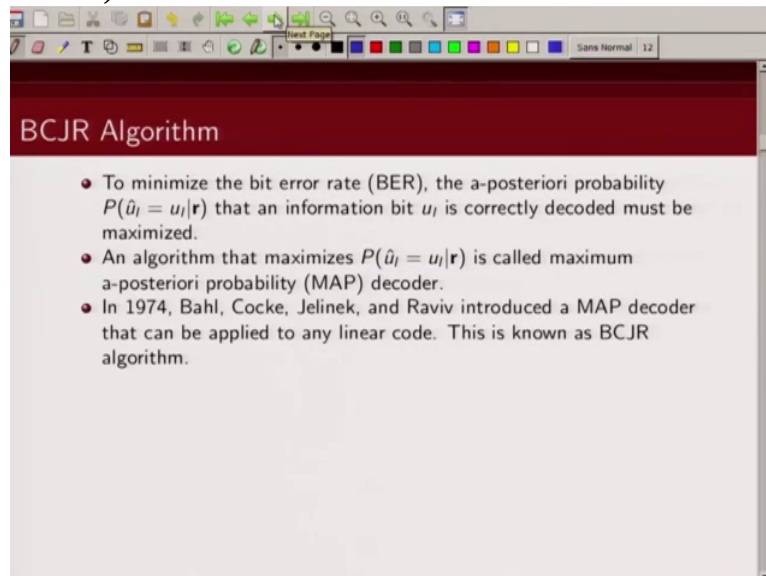
So we are going to use a posteriori probability based algorithm to estimate our information sequence. And this

(Refer Slide Time 01:25)



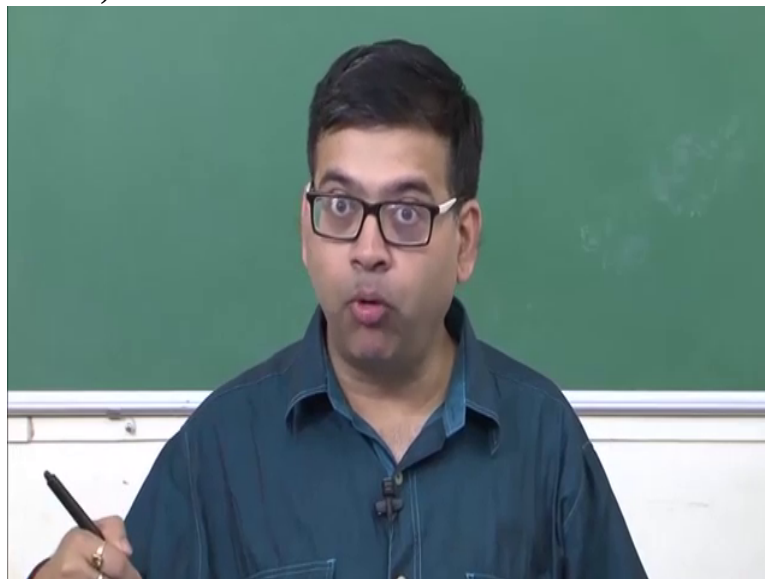
algorithm which maximizes probability of \hat{u} given u given the received sequence \mathbf{r} is known as MAP decoder. Now this is known as, also known as

(Refer Slide Time 01:41)



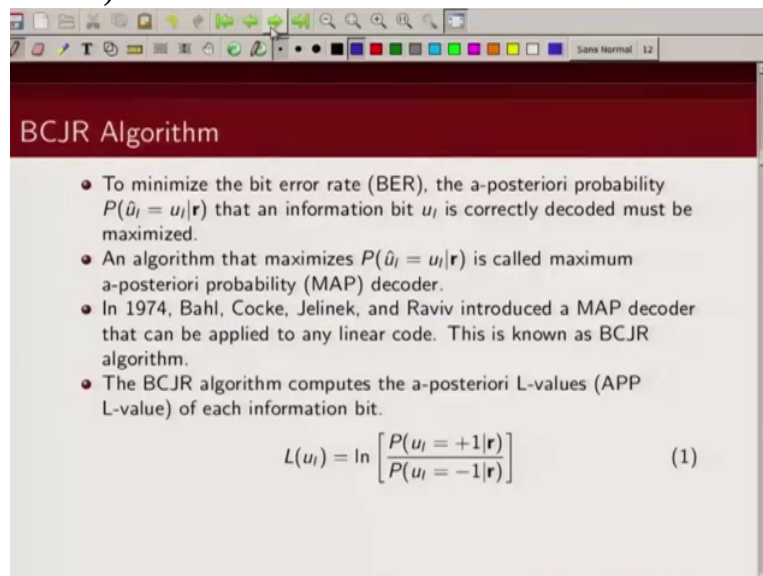
B C J R algorithm named after these researchers who, Bahl, Cocke, Jelinek and Raviv, who introduced this algorithm in 1974. And this algorithm can be applied to any linear code, block code or

(Refer Slide Time 01:58)



convolutional code.

(Refer Slide Time 02:02)



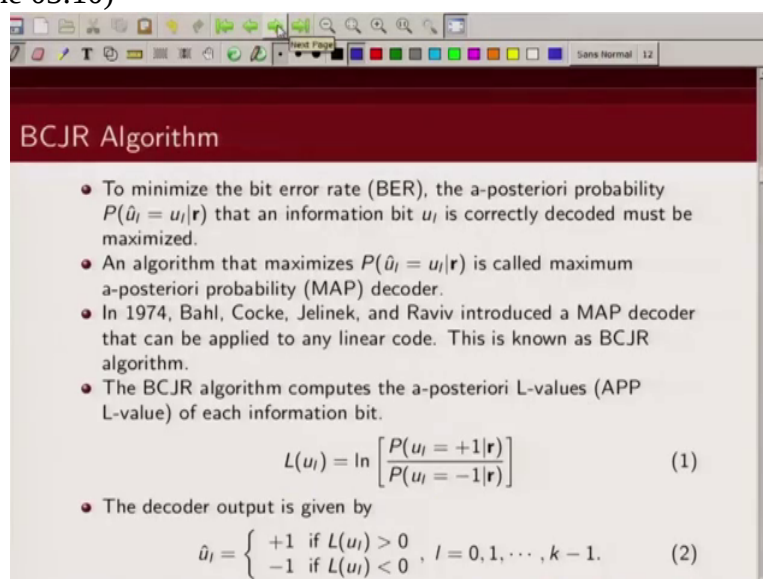
The slide is titled "BCJR Algorithm" and contains the following text:

- 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.

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

Now the complexity of this algorithm was much higher than Viterbi algorithm and that's why it was not popular in 70s, but in late 90s when, this concatenated codes, turbo codes came into picture and we required soft estimates then these algorithms became very, very popular. So what this algorithm does, it computes the a posteriori probability. So I define a posteriori, Log-likelihood value, I call it L value like this. So it basically computes probability of u l being plus 1 given a received sequence r by probability of u l being minus 1 given recieved sequence r. Take a log of that. Now if this L value is greater than zero, then you decide in favor of u l being plus 1, otherwise you decide in favor of u l being minus 1.

(Refer Slide Time 03:10)



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

- 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 plus 1 if the L value is greater than 0, otherwise you decide in favor of minus 1. So we are now going to talk about how to compute these terms, these terms you see in computation of

(Refer Slide Time 03:31)

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

- 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

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

A P P value, how do we compute these terms and how we can

(Refer Slide Time 03:37)

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

- 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.

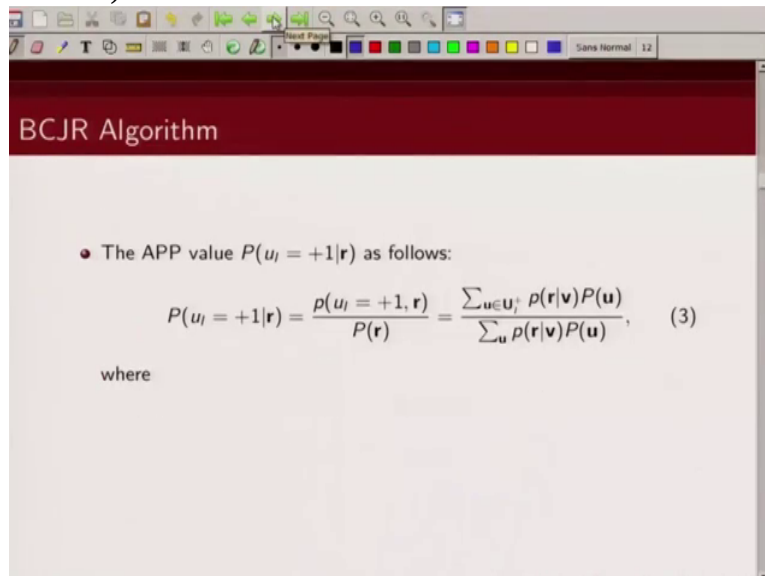
$$L(u_l) = \ln \left[\frac{P(u_l = +1 | \mathbf{r})}{P(u_l = -1 | \mathbf{r})} \right] \quad (1)$$

- The decoder output is given by

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

exploit the structure of the Trellis of the convolutional encoder to simplify this expression.

(Refer Slide Time 03:46)



BCJR Algorithm

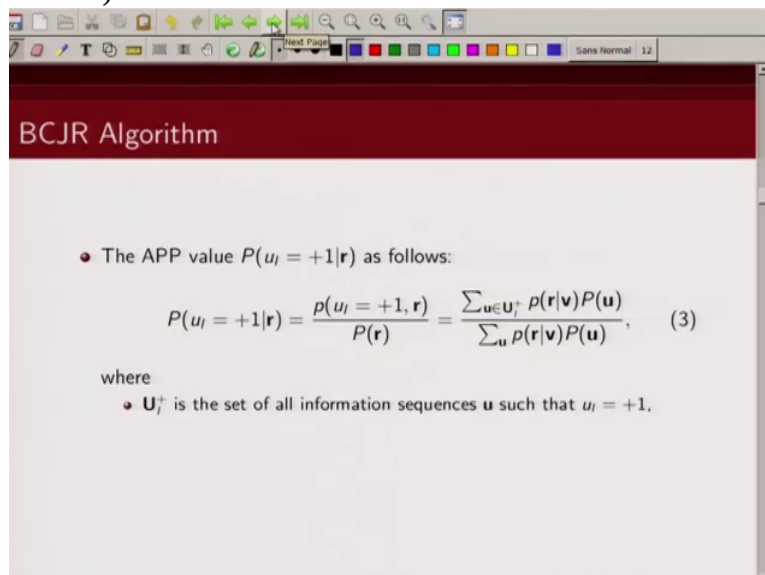
- The APP value $P(u_i = +1|\mathbf{r})$ as follows:

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

where

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

(Refer Slide Time 04:34)



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 $u_i = +1$.

So as I said, since we are interested in joint probability of u_i being plus 1 and \mathbf{r} we sum this probability over all those set of

(Refer Slide Time 04:47)

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 $u_i = +1$.

information sequences where the bit, the corresponding bit is plus 1.

(Refer Slide Time 04:57)

BCJR Algorithm

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

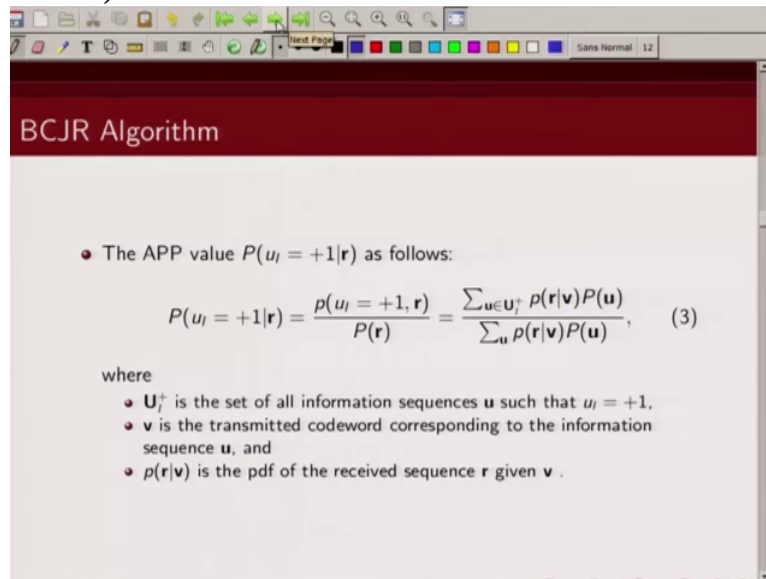
$$P(u_i = +1 | \mathbf{r}) = \frac{p(u_i = +1, \mathbf{r})}{P(\mathbf{r})} = \frac{\sum_{\mathbf{u} \in \mathbf{U}_i^+} p(\mathbf{r} | \mathbf{v}) P(\mathbf{u})}{\sum_{\mathbf{u}} p(\mathbf{r} | \mathbf{v}) P(\mathbf{u})}, \quad (3)$$

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

And our transmitted codeword is \mathbf{v} , our information sequence is \mathbf{u} and \mathbf{r} is the received sequence. Probability

(Refer Slide Time 05:07)



BCJR Algorithm

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

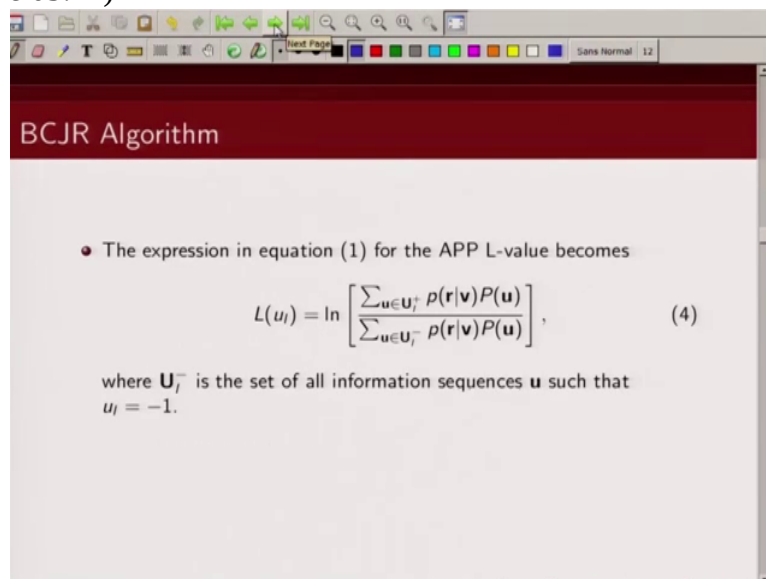
$$P(u_i = +1|\mathbf{r}) = \frac{p(u_i = +1, \mathbf{r})}{P(\mathbf{r})} = \frac{\sum_{\mathbf{u} \in \mathbf{U}_i^+} p(\mathbf{r}|\mathbf{v})P(\mathbf{u})}{\sum_{\mathbf{u}} p(\mathbf{r}|\mathbf{v})P(\mathbf{u})}, \quad (3)$$

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} .

of \mathbf{r} given \mathbf{v} can be computed from the channel, given channel.

(Refer Slide Time 05:14)



BCJR Algorithm

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

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

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

Similarly we can also compute, now if you go back here, the denominator we need to compute probability of

(Refer Slide Time 05:24)

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

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

u_l being minus 1 given \mathbf{r} so similar to this term we can also write probability of u_l being

(Refer Slide Time 05:32)

BCJR Algorithm

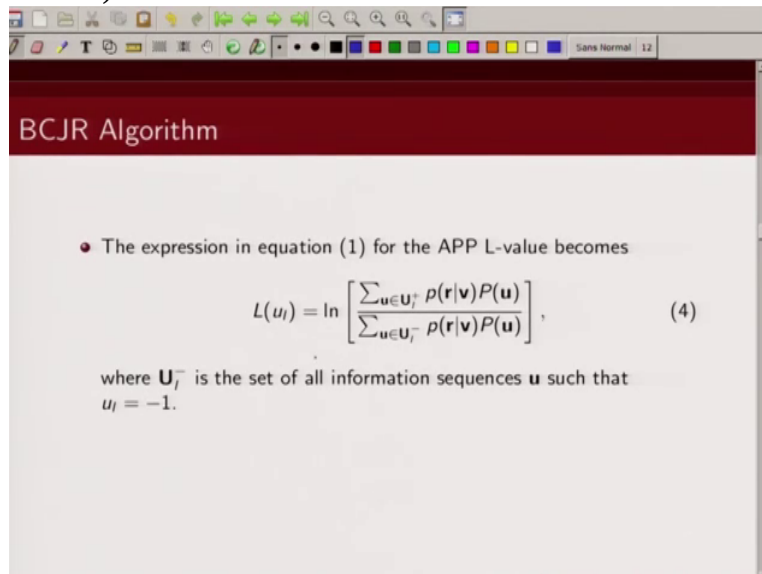
- The APP value $P(u_l = +1 | \mathbf{r})$ as follows:

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

where

minus 1 given \mathbf{r} . And probability of \mathbf{r} is a common term. So if we do that,

(Refer Slide Time 05:40)



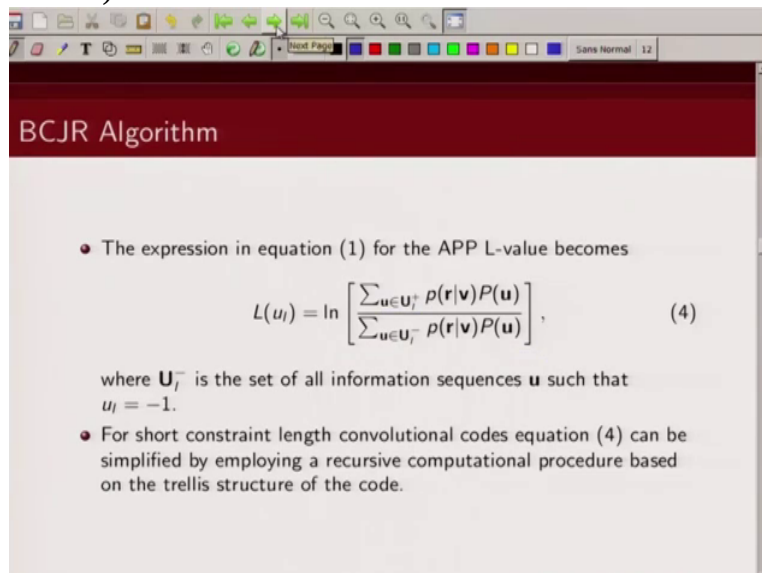
The expression in equation (1) for the APP L-value becomes

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

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

what we get is this. So again this L value, the APP value of u_l is given by probability of \mathbf{r} given \mathbf{v} multiplied by probability of \mathbf{u} where we are summing over all information sequences where the corresponding bit is plus 1. And similarly for the denominator we are summing over all information sequences where information bit is minus 1. We will illustrate this with the help of example and then things will be little more clear.

(Refer Slide Time 06:26)



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

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

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

- 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 note here, if you have very large sequences, this is sum over all input sequences where u_l is plus 1 and this is sum over all input sequences where u_l is minus 1. So if your information sequence is large this is sum over very large number of possibilities. So this is quite complex.

(Refer Slide Time 06:49)



Now can we use the structure of the convolutional code to simplify this expression?

(Refer Slide Time 06:56)

A screenshot of a presentation slide titled "BCJR Algorithm". The slide contains a bullet point, a mathematical equation, a definition of a set, and another bullet point. The equation is labeled (4). The slide is shown within a window with a standard operating system toolbar at the top.

BCJR Algorithm

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

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

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

- 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 4 by using the Trellis structure of

(Refer Slide Time 07:08)



the convolutional code. We know all possible transitions are not possible. So our Trellis diagram or the state diagram will, will ensure, will tell us what are the valid transitions. So we can simplify

(Refer Slide Time 07:22)

A screenshot of a presentation slide titled "BCJR Algorithm". The slide contains a bullet point, a mathematical equation, and another bullet point. The equation is labeled (4). The slide is displayed in a window with a standard toolbar at the top.

BCJR Algorithm

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

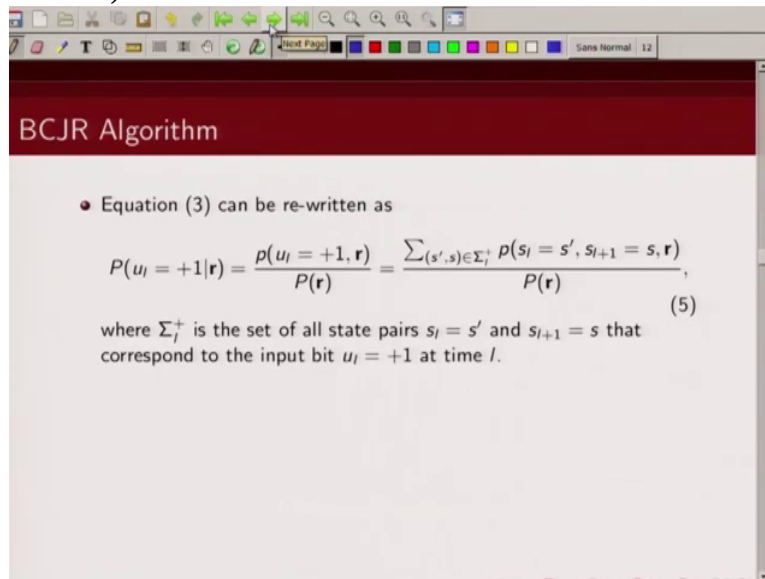
$$L(u_i) = \ln \left[\frac{\sum_{\mathbf{u} \in \mathbf{U}_i^+} \rho(\mathbf{r}|\mathbf{v})P(\mathbf{u})}{\sum_{\mathbf{u} \in \mathbf{U}_i^-} \rho(\mathbf{r}|\mathbf{v})P(\mathbf{u})} \right], \quad (4)$$

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.

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 07:42)



BCJR Algorithm

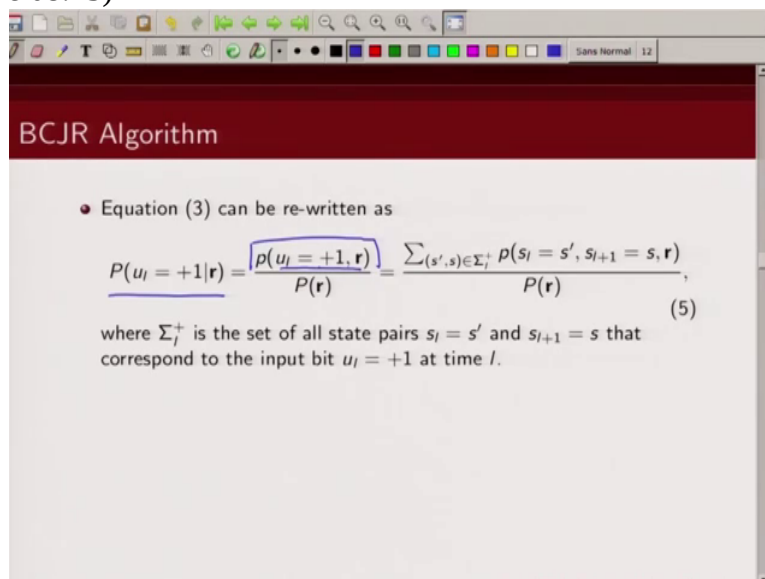
- Equation (3) can be re-written as

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

where Σ_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. We again go back and look at this probability of this u_l being plus 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 probability of \mathbf{r} . Now we are going to, now look at this expression. This is joint probability of u_l being plus 1 and

(Refer Slide Time 08:13)



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 Σ_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 .

given the received sequence \mathbf{r} has been received. So if you look at any Trellis diagram, let's say this is some Trellis diagram, simple 2 state code, like that you have, so we are interested in where u_l is

(Refer Slide Time 08:38)

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 Σ_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 .

plus 1 and where u_l is minus 1. Let us say this is 0 by 0 0, this is 1 by 1 1, this is, let's say 1 by 1 0, this is 0 by 0 1.

(Refer Slide Time 08:52)

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 Σ_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's look at one Trellis section. So we are interested in all those transitions which belongs to u_l plus 1. Now what are those transitions? So in this example this is one such

(Refer Slide Time 09:07)

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 Σ_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 .

transition. And the other is this transition,

(Refer Slide Time 09:11)

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 Σ_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 .

Ok. So what I am writing here is then I am interested in what's the joint probability that the previous state is s prime, the next state is s and the recieved sequence is \mathbf{r} and I am summing over all those

(Refer Slide Time 09:35)

BCJR Algorithm

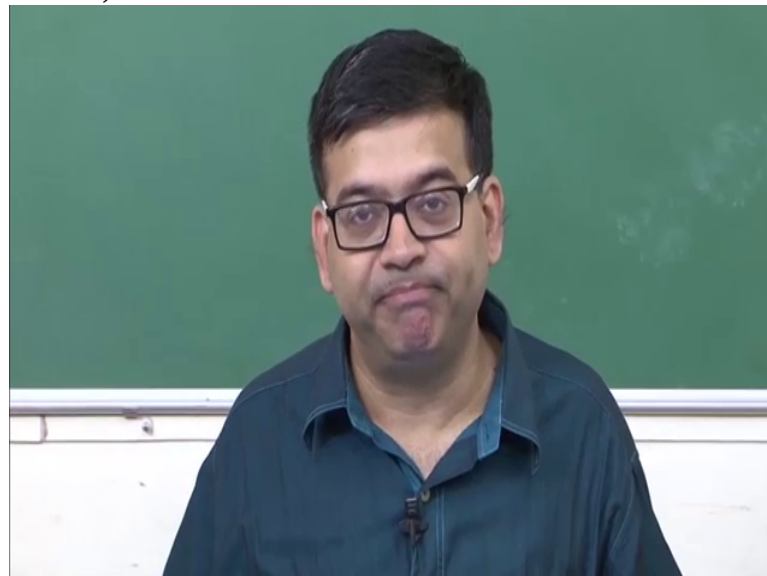
- Equation (3) can be re-written as

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

where Σ_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 .

state transitions that belong to the set pair where the input corresponds to this transition is plus 1. So note what is my this sigma l plus, it is a set of all state pairs where the initial state is s prime then next state is s so its, it's a pair of states where the transitions, the input bit corresponding to a valid transition is plus 1. So, so in this case the set that belongs to this is given by this red line, Ok.

(Refer Slide Time 10:24)



So I can write the joint probability of u l being plus 1 and r in terms

(Refer Slide Time 10:31)

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 Σ_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 .

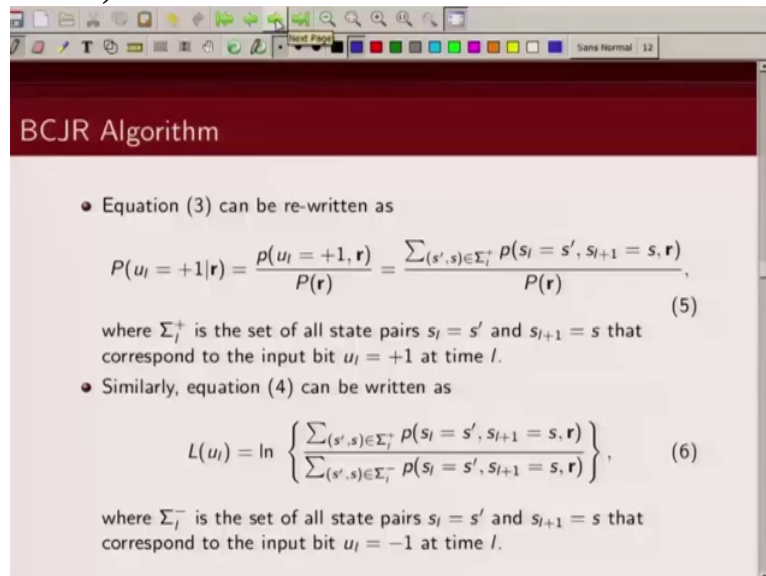
of condition on the valid Trellis transitions in this way, I can write it as what is the probability that the initial state is s prime, next state is s given the received sequence r and I sum over all those transitions which belong to input bit being plus 1.

(Refer Slide Time 10:56)



Similarly

(Refer Slide Time 11:00)



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

- 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 Σ_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 Σ_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 .

I can write exactly the way I wrote, probability of u_l being plus 1 given \mathbf{r} , I can follow the same procedure to write what is the probability of u_l being minus 1 given \mathbf{r} . So what would be the change here? So I will compute this probability and I will sum over all those state pairs which correspond to

(Refer Slide Time 11:25)



input bit minus 1. So if I plug

(Refer Slide Time 11:30)

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 Σ_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 Σ_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 .

these values of probabilities which are given by equation 5 and similarly I can calculate the probability of u l being minus 1 given r so instead of this thing here

(Refer Slide Time 11:45)

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 Σ_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 Σ_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 .

I will have summation over s prime as summation over all those pairs which corresponds to u l being minus 1

(Refer Slide Time 11:54)

BCJR Algorithm

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

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

where Σ_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 Σ_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 .

and I will get this same thing here. So if I do that what I will get is equation number 6. So note that previously I had the same expression, equation number 4

(Refer Slide Time 12:10)

BCJR Algorithm

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

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

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

- 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.

in terms of this input sequence u_l . Now if our

(Refer Slide Time 12:17)



input sequence is very long this is summation over a large number of

(Refer Slide Time 12:20)

A screenshot of a presentation slide titled "BCJR Algorithm". The slide has a red header with the title in white. 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 shown within a window with a standard toolbar at the top.

terms where as I have now

(Refer Slide Time 12:23)

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 Σ_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 .

simplified my expression.

(Refer Slide Time 12:25)

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 Σ_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 Σ_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 .

Handwritten notes: $p(u_l = -1 | \mathbf{r})$ and $\Sigma_{(s', s)}^-$

So this, the summation is now only over valid transitions corresponding to u_l being plus 1 and this summation is over valid transitions corresponding to u_l being minus 1.

(Refer Slide Time 12:41)



So I have simplified my equation number 4

(Refer Slide Time 12:45)

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 Σ_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 Σ_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 .

in equation number 6 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 likelihood 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's 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 prime, the next state is s given a received sequence r? So as I said

(Refer Slide Time 13:35)

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 .

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 equal to 1, t equal to 2 so this is my let us say time instances and I get some bits, let us say I get some

(Refer Slide Time 13:56)

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 .

r_1 corresponds to what I receive at time t equal to 1, r_2 corresponds to what I receive at time 2, r_l corresponds to what I receive in time l and like that, r_{l+1} is what I receive at time t equal to $l+1$, like that. So this

(Refer Slide Time 14:17)

BCJR Algorithm

$t_1 \quad t_2 \quad t_k \quad t_{k+1}$

- 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 .

received is, whole thing is my received sequence r , Ok. Now what I am doing is I will partition that received sequence into 3 segments. So one, which corresponds to, one is this,

(Refer Slide Time 14:36)

BCJR Algorithm

$t_1 \quad t_2 \quad t_k \quad t_{k+1}$

- 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 .

which corresponds to time before l . So one is this portion, this portion of my received sequence. This is $r_{t < l}$. Next

(Refer Slide Time 14:50)

BCJR Algorithm

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

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

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

is this section which corresponds to r_t greater than t and then

(Refer Slide Time 14:58)

BCJR Algorithm

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

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

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

third section is this, which corresponds to r_t , Ok. So what I did was I split this r into 3 segments. One is r corresponds to time less than t , r_t at time t and

(Refer Slide Time 15:18)

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 .

r at time greater than

(Refer Slide Time 15:21)

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 .

I. Now using base rule I can write this probability as probability of r at time greater than l given s prime and s and this r into probability of s prime s and r at t less than l and r l. Now subsequently I can further simplify this, again apply Bayes rule and I can write

(Refer Slide Time 15:54)

this as probability of s and r l given s prime and r t less than l into probability of s prime and r t less than l. So note now this term that I had here, so applying Bayes rule essentially,

(Refer Slide Time 16:17)

I broke it up into 3 terms. One is this term, second is this term and third is this term, Ok. Now let's look at this. So probability of r when t is greater than l given initial state s prime, next state s and the received sequence before l and received sequence is l. So let us look at the Trellis diagram. Let's go back and look at the Trellis diagram at time l. Let's take this example of 2 state code. So what I had was 0 by 0 0, 1 by 1 1 then I had this, 1 as 1 0 and this was 0 by 0 1. So this was my Trellis diagram. This is all zero state; this is state 1, Ok.

(Refer Slide Time 17:16)

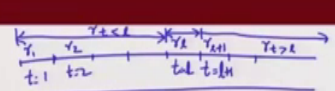
Now note and like that you have, you have, in Trellis diagram you have, this is one Trellis section. You will similarly have Trellis sections others. So this is a time l. So you are interested in what is the probability of r t greater than l given previous state s prime given next state s given the received sequence before time

(Refer Slide Time 17:45)

t equal to l and given the current received sequence. Now note that if I specify this next state, so probability of r t greater than l given s then I don't need information about the previous state. I don't need information about what is the current input, I don't

(Refer Slide Time 18:12)

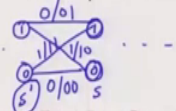
BCJR Algorithm



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

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

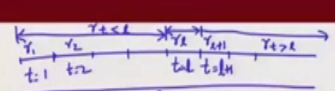
where $r_{t<l}$ represents the portion of the received sequence r before time l and $r_{t>l}$ represents the portion of the received sequence r after time l .



need information about what was the received

(Refer Slide Time 18:15)

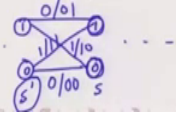
BCJR Algorithm



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

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

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 .



sequence before l provided I know what is the next state s .

(Refer Slide Time 18:21)

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 this probability that you see here, probability of $r_{t>l}$

(Refer Slide Time 18:28)

BCJR Algorithm

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

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

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 .

than l given s prime s and this received sequence r can be then written as probability of $r_{t>l}$ greater than l given only s because knowing this final state s I don't need information about what was my state

(Refer Slide Time 18:47)

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 .

here. I don't need information about what my received sequence was here. I don't need information about what my past received sequence was, provided I know what was my next state s . So this, given these quantities will only depend on s . So I can simplify this expression like this. The same thing here, look at probability of being s at l given previous state and given the input before time t equal to l . Now if I specify what the previous state is, then I don't

(Refer Slide Time 19:25)

BCJR Algorithm

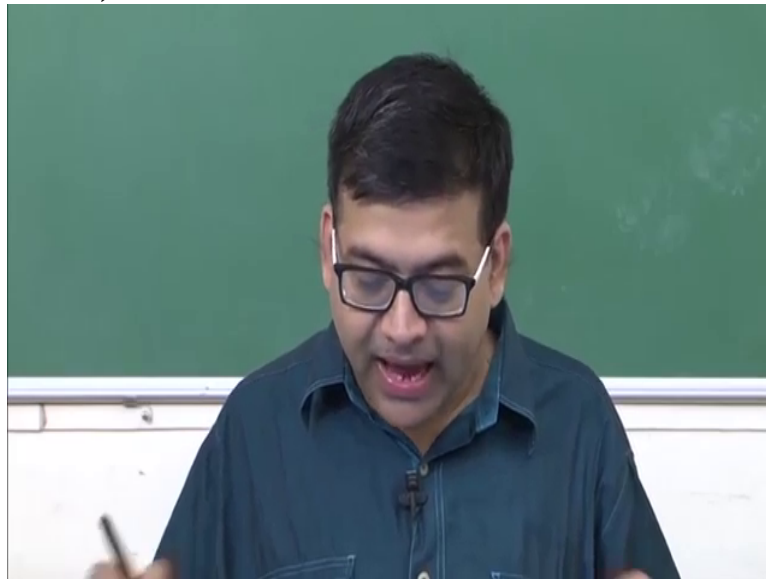
The joint pdf's $p(s', s, r)$ in equation (6) can be evaluated recursively

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

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 .

need what was my input at time t less than l . 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 joint probability we have now split up into three probabilities, one is this, second one is this, and third one is this, Ok

(Refer Slide Time 19:57)



and we will now show how we can compute each of these

(Refer Slide Time 20:02)

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 .

terms. So let us call

(Refer Slide Time 20:05)

BCJR Algorithm

- Defining
 - $\alpha_l(s') \equiv p(s', r_{t < l})$ (8)
 - $\gamma_l(s', s) \equiv p(s, r_l | s')$ (9)
 - $\beta_{l+1}(s) \equiv p(r_{t > l} | s)$ (10)

this probability by alpha, this probability by gamma and this probability by beta. And now we are going to show then we can write then this joint

(Refer Slide Time 20:20)

BCJR Algorithm

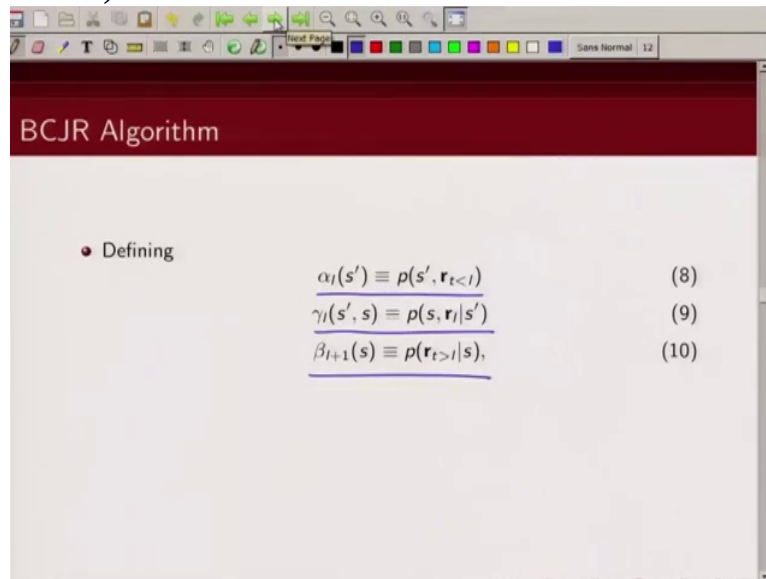
The joint pdf's $p(s', s, r)$ in equation (6) can be evaluated recursively

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

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 .

probability in terms alpha, beta and gamma.

(Refer Slide Time 20:25)



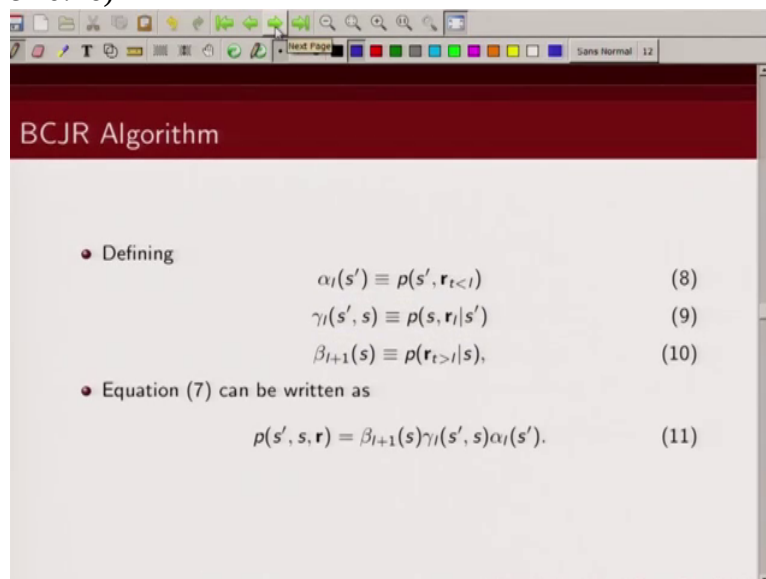
BCJR Algorithm

- Defining

$$\alpha_l(s') \equiv p(s', \mathbf{r}_{t < l}) \quad (8)$$
$$\gamma_l(s', s) \equiv p(s, \mathbf{r}_l | s') \quad (9)$$
$$\beta_{l+1}(s) \equiv p(\mathbf{r}_{t > l} | s), \quad (10)$$

So

(Refer Slide Time 20:26)



BCJR Algorithm

- Defining

$$\alpha_l(s') \equiv p(s', \mathbf{r}_{t < l}) \quad (8)$$
$$\gamma_l(s', s) \equiv p(s, \mathbf{r}_l | s') \quad (9)$$
$$\beta_{l+1}(s) \equiv p(\mathbf{r}_{t > l} | s), \quad (10)$$

- Equation (7) can be written as

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

we can now write our equations in terms of alpha, beta and

(Refer Slide Time 20:32)

BCJR Algorithm

- Defining
$$\alpha_l(s') \equiv p(s', \mathbf{r}_{t < l}) \quad (8)$$
$$\gamma_l(s', s) \equiv p(s, \mathbf{r}_l | s') \quad (9)$$
$$\beta_{l+1}(s) \equiv p(\mathbf{r}_{t > l} | s), \quad (10)$$
- Equation (7) can be written as
$$p(s', s, \mathbf{r}) = \beta_{l+1}(s) \gamma_l(s', s) \alpha_l(s'). \quad (11)$$

gamma, Ok. So let us now talk about how we can compute alpha, beta and gamma. So these

(Refer Slide Time 20:46)

BCJR Algorithm

- The expression for the probability $\alpha_{l+1}(s)$ can now be rewritten as
$$\begin{aligned} \alpha_{l+1}(s) &= p(s, \mathbf{r}_{t < l+1}) = \sum_{s' \in \sigma_l} p(s', s, \mathbf{r}_{t < l+1}) \\ &= \sum_{s' \in \sigma_l} p(s, \mathbf{r}_l | s') p(s', \mathbf{r}_{t < l}) \\ &= \sum_{s' \in \sigma_l} p(s, \mathbf{r}_l | s') p(s', \mathbf{r}_{t < l}) \\ &= \sum_{s' \in \sigma_l} \gamma_l(s', s) \alpha_l(s'), \end{aligned} \quad (12)$$

where σ_l is the set of all states at time l .

alphas can be computed using forward recursion as follows. So let us look at what is alpha plus 1 s. Now go back to our definition. So probability,

(Refer Slide Time 20:59)

BCJR Algorithm

- Defining

$$\alpha_l(s') \equiv p(s', \mathbf{r}_{t < l}) \quad (8)$$

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

$$\beta_{l+1}(s) \equiv p(\mathbf{r}_{t > l} | s), \quad (10)$$
- Equation (7) can be written as

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

joint probability of being in state s prime and received sequence at time t less than l.

(Refer Slide Time 21:09)

BCJR Algorithm

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

$$\begin{aligned} \alpha_{l+1}(s) &= p(s, \mathbf{r}_{t < l+1}) = \sum_{s' \in \sigma_l} p(s', s, \mathbf{r}_{t < l+1}) \\ &= \sum_{s' \in \sigma_l} p(s, \mathbf{r}_l | s', \mathbf{r}_{t < l}) p(s', \mathbf{r}_{t < l}) \\ &= \sum_{s' \in \sigma_l} p(s, \mathbf{r}_l | s') p(s', \mathbf{r}_{t < l}) \\ &= \sum_{s' \in \sigma_l} \gamma_l(s', s) \alpha_l(s'), \end{aligned} \quad (12)$$

where σ_l is the set of all states at time l .

So alpha l plus 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, previous state s prime and summing over all previous state. So what is this, summation over s prime belonging to all possible state at time l? So what I did was I had some term, probability term, probability of let's 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's what I did here. I introduced a new variable

(Refer Slide Time 21:55)

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

$$\begin{aligned} \alpha_{l+1}(s) &= p(s, \mathbf{r}_{t < l+1}) = \sum_{s' \in \sigma_l} p(s', s, \mathbf{r}_{t < l+1}) \\ &= \sum_{s' \in \sigma_l} p(s, \mathbf{r}_l | s', \mathbf{r}_{t < l}) p(s', \mathbf{r}_{t < l}) \\ &= \sum_{s' \in \sigma_l} p(s, \mathbf{r}_l | s') p(s', \mathbf{r}_{t < l}) \\ &= \sum_{s' \in \sigma_l} \gamma_l(s', s) \alpha_l(s'), \end{aligned} \quad (12)$$

where σ_l is the set of all states at time l .

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, this is following exactly the same procedure which

(Refer Slide Time 22:16)

Defining

$$\alpha_l(s') \equiv p(s', \mathbf{r}_{t < l}) \quad (8)$$

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

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

Equation (7) can be written as

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

we followed

(Refer Slide Time 22:17)

BCJR Algorithm

- Defining

$$\alpha_l(s') \equiv p(s', r_{t < l}) \quad (8)$$

$$\gamma_l(s', s) \equiv p(s, r_l | s') \quad (9)$$

$$\beta_{l+1}(s) \equiv p(r_{t > l} | s), \quad (10)$$

here.

(Refer Slide Time 22:19)

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 .

When we wrote this, we are basically using Bayes rule, now using Bayes rule, I can write this probability as product of these two probabilities. Now again the probability of s and r_l given s' and received sequence at time t less than l , if you know the previous state s' you don't need

(Refer Slide Time 22:49)

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

$$\begin{aligned} \alpha_{l+1}(s) &= \frac{p(s, \mathbf{r}_{t < l+1})}{\sum_{s' \in \sigma_l} p(s', \mathbf{r}_{t < l+1})} \\ &= \sum_{s' \in \sigma_l} \frac{p(s, \mathbf{r}_l | s') p(s', \mathbf{r}_{t < l})}{\sum_{c \in \mathcal{C}} p(a, b, c)} \\ &= \sum_{s' \in \sigma_l} p(s, \mathbf{r}_l | s') p(s', \mathbf{r}_{t < l}) \\ &= \sum_{s' \in \sigma_l} \gamma_l(s', s) \alpha_l(s'), \end{aligned} \quad (12)$$

where σ_l is the set of all states at time l .

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

(Refer Slide Time 23:04)

Defining

$$\alpha_l(s') \equiv p(s', \mathbf{r}_{t < l}) \quad (8)$$

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

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

Equation (7) can be written as

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

our alpha and what is the next term, this is our gamma. So what I have shown you here

(Refer Slide Time 23:11)

BCJR Algorithm

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

$$\begin{aligned}
 \alpha_{l+1}(s) &= \frac{p(s, r_{t < l+1})}{\sum_{s' \in \sigma_l} p(s', r_{t < l+1})} = \frac{p(s, r_l | s', r_{t < l}) p(s', r_{t < l})}{\sum_{s' \in \sigma_l} p(s, r_l | s', r_{t < l}) p(s', r_{t < l})} \quad \begin{matrix} p(a,b) \\ \sum_c p(a,b,c) \end{matrix} \\
 &= \frac{p(s, r_l | s', r_{t < l}) p(s', r_{t < l})}{\sum_{s' \in \sigma_l} p(s, r_l | s', r_{t < l}) p(s', r_{t < l})} \\
 &= \frac{p(s, r_l | s') p(s', r_{t < l})}{\sum_{s' \in \sigma_l} p(s, r_l | s') p(s', r_{t < l})} \\
 &= \sum_{s' \in \sigma_l} \gamma_l(s', s) \alpha_l(s'), \quad (12)
 \end{aligned}$$

where σ_l is the set of all states at time l .

then is alpha at next state s

(Refer Slide Time 23:16)



can be written as, can be computed recursively from alphas at previous state in this particular fashion. So again

(Refer Slide Time 23:27)

BCJR Algorithm

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

$$\begin{aligned}
 \alpha_{l+1}(s) &= \frac{p(s, r_{t < l+1})}{\sum_{s' \in \sigma_l} p(s', r_{t < l+1})} = \sum_{s' \in \sigma_l} \frac{p(s, r_l | s', r_{t < l}) p(s', r_{t < l})}{\sum_{c} p(a, b, c)} \\
 &= \sum_{s' \in \sigma_l} \frac{p(s, r_l | s') p(s', r_{t < l})}{\sum_{c} p(a, b, c)} \\
 &= \sum_{s' \in \sigma_l} \gamma_l(s', s) \alpha_l(s'), \tag{12}
 \end{aligned}$$

where σ_l is the set of all states at time l .

let's illustrate this with an example. Let's go back to our 2 state code example. So this is 2 state code. This is my all zero state. This is state 1, there are 2 transitions. Let's say this is 0 input, output 0 0, input 1, output 1 1,

(Refer Slide Time 23:49)

BCJR Algorithm

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

$$\begin{aligned}
 \alpha_{l+1}(s) &= \frac{p(s, r_{t < l+1})}{\sum_{s' \in \sigma_l} p(s', r_{t < l+1})} = \sum_{s' \in \sigma_l} \frac{p(s, r_l | s', r_{t < l}) p(s', r_{t < l})}{\sum_{c} p(a, b, c)} \\
 &= \sum_{s' \in \sigma_l} \frac{p(s, r_l | s') p(s', r_{t < l})}{\sum_{c} p(a, b, c)} \\
 &= \sum_{s' \in \sigma_l} \gamma_l(s', s) \alpha_l(s'), \tag{12}
 \end{aligned}$$

where σ_l is the set of all states at time l .

here input 1, output 1 0 and here input 0 and output 0 1.

(Refer Slide Time 23:57)

BCJR Algorithm

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

$$\begin{aligned} \alpha_{l+1}(s) &= \frac{p(s, r_{t < l+1})}{\sum_{s' \in \sigma_l} p(s', r_{t < l+1})} = \frac{p(s, r_l | s') p(s', r_{t < l})}{\sum_{s' \in \sigma_l} p(s, r_l | s') p(s', r_{t < l})} \quad \begin{matrix} p(a, b) \\ \sum_c p(a, b, c) \end{matrix} \\ &= \sum_{s' \in \sigma_l} \frac{p(s, r_l | s') p(s', r_{t < l})}{\sum_{s' \in \sigma_l} p(s, r_l | s') p(s', r_{t < l})} \\ &= \sum_{s' \in \sigma_l} \gamma_l(s', s) \alpha_l(s'), \end{aligned} \quad (12)$$

where σ_l is the set of all states at time l .

So then what would be the value of alpha? So let's say this is time t equal to some l and this is time t equal to l plus 1. So

(Refer Slide Time 24:10)

BCJR Algorithm

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

$$\begin{aligned} \alpha_{l+1}(s) &= \frac{p(s, r_{t < l+1})}{\sum_{s' \in \sigma_l} p(s', r_{t < l+1})} = \frac{p(s, r_l | s') p(s', r_{t < l})}{\sum_{s' \in \sigma_l} p(s, r_l | s') p(s', r_{t < l})} \quad \begin{matrix} p(a, b) \\ \sum_c p(a, b, c) \end{matrix} \\ &= \sum_{s' \in \sigma_l} \frac{p(s, r_l | s') p(s', r_{t < l})}{\sum_{s' \in \sigma_l} p(s, r_l | s') p(s', r_{t < l})} \\ &= \sum_{s' \in \sigma_l} \gamma_l(s', s) \alpha_l(s'), \end{aligned} \quad (12)$$

where σ_l is the set of all states at time l .

how can we write let's say alpha at l plus 1 for the state 0?

(Refer Slide Time 24:17)

BCJR Algorithm

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

$$\alpha_{l+1}(s) = \frac{p(s, r_{t < l+1})}{\sum_{s' \in \sigma_l} p(s', r_{t < l+1})} = \sum_{s' \in \sigma_l} \frac{p(s, r_l | s', r_{t < l}) p(s', r_{t < l})}{\sum_{s' \in \sigma_l} p(s, r_l | s') p(s', r_{t < l})} \gamma_l(s', s) \alpha_l(s')$$

where σ_l is the set of all states at time l .

Now note here this is given by product of this summation over all input state right, now so alpha l 0 can be written as then gamma, this can be written as gamma at time l of 0 0. gamma l 0 0 is previous state is 0, next state is 0, gamma 0 0 into alpha l belonging to state 0. So this is gamma l 0 0, alpha l 0 so this is corresponding to this transition, Ok. This is corresponding to this transition, this term will come, fine. Now there is another transition here which is basically this. So we can write this will be plus gamma l 1 0 so gamma l 1 0 is the gamma corresponding to this transition when the initial state is 1 and next state is 0 multiplied by alpha at time l belonging to state 1, Ok . So alpha l plus l 0 can be then written

(Refer Slide Time 25:53)

BCJR Algorithm

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

$$\alpha_{l+1}(s) = \frac{p(s, r_{t < l+1})}{\sum_{s' \in \sigma_l} p(s', r_{t < l+1})} = \sum_{s' \in \sigma_l} \frac{p(s, r_l | s', r_{t < l}) p(s', r_{t < l})}{\sum_{s' \in \sigma_l} p(s, r_l | s') p(s', r_{t < l})} \gamma_l(s', s) \alpha_l(s')$$

where σ_l is the set of all states at time l .

as this. Now similarly we can also compute what is the value of

(Refer Slide Time 26:01)

BCJR Algorithm

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

$$\alpha_{l+1}(s) = \frac{p(s, r_{t < l+1})}{\sum_{s' \in \sigma_l} p(s', r_{t < l+1})} = \sum_{s' \in \sigma_l} \frac{p(s, r_l | s', r_{t < l}) p(s', r_{t < l})}{\sum_{s' \in \sigma_l} p(s, r_l | s') p(s', r_{t < l})} \gamma_l(s', s) \alpha_l(s')$$

where σ_l is the set of all states at time l .

alpha l plus 1 at 1. So we repeat the same procedure. So let's write it here. alpha l plus 1 in the final state is 1 can be written as gamma l 0 1 times alpha l 0 plus, so this is corresponding to this transition, gamma l initial state 0, final

(Refer Slide Time 26:35)

BCJR Algorithm

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

$$\alpha_{l+1}(s) = \frac{p(s, r_{t < l+1})}{\sum_{s' \in \sigma_l} p(s', r_{t < l+1})} = \sum_{s' \in \sigma_l} \frac{p(s, r_l | s', r_{t < l}) p(s', r_{t < l})}{\sum_{s' \in \sigma_l} p(s, r_l | s') p(s', r_{t < l})} \gamma_l(s', s) \alpha_l(s')$$

where σ_l is the set of all states at time l .

state 1 and alpha at time l 0 plus this another transition which is this. So this can be written as gamma l 1 1 times alpha l 1. So these are, for particular convolutional encoder

(Refer Slide Time 27:03)

BCJR Algorithm

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

$$\alpha_{l+1}(s) = \frac{p(s, r_{t < l+1})}{\sum_{s' \in \sigma_l} p(s', r_{t < l+1})} = \sum_{s' \in \sigma_l} \frac{p(s, r_l | s', r_{t < l}) p(s', r_{t < l})}{\sum_{c} p(a, b, c)}$$

$$= \sum_{s' \in \sigma_l} p(s, r_l | s') p(s', r_{t < l})$$

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

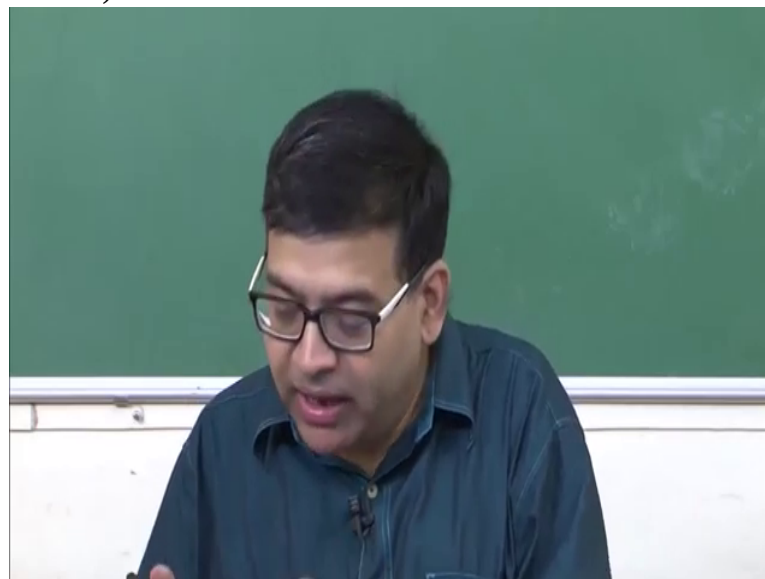
where σ_l is the set of all states at time l .

Handwritten notes on the slide:

- $\alpha_{s_{l+1}}(0) = \gamma_x(0,0) \alpha_x(0) + \gamma_x(1,0) \alpha_x(1)$
- $\alpha_{s_{l+1}}(1) = \gamma_x(0,1) \alpha_x(0) + \gamma_x(1,1) \alpha_x(1)$

whose Trellis section is given by this, these are, these 2 are my alpha values, this one and this. So you see I can recursively compute alpha time

(Refer Slide Time 27:17)



l plus 1 from alpha at time l and branch metric gamma. Now to do this recursion, we need to know what is the initial condition. What is the initial condition? We need to know

(Refer Slide Time 27:35)

BCJR Algorithm

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

$$\alpha_{l+1}(s) = \frac{p(s, r_{t < l+1})}{\sum_{s' \in \sigma_l} p(s', r_{t < l+1})} = \sum_{s' \in \sigma_l} \frac{p(s, r_l | s', r_{t < l}) p(s', r_{t < l})}{\sum_{c} p(a, b, c)}$$

$$= \sum_{s' \in \sigma_l} p(s, r_l | s') p(s', r_{t < l})$$

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

where σ_l is the set of all states at time l .

Handwritten notes and diagram:

- Left box: $\alpha_{2H}(0) = \gamma_x(0,0)\alpha_x(0) + \gamma_x(1,0)\alpha_x(1)$
- Right box: $\alpha_{2H}(j) = \gamma_x(0,j)\alpha_x(0) + \gamma_x(1,j)\alpha_x(1)$
- Diagram: A trellis with states 0 and 1 at time $t=l$ and $t=l+1$. Transitions are labeled with bit pairs: 0/01, 1/10, 0/00, 1/11.

what is the value of alpha 0 for different states, for state 0, for state 1. We need to know what the values of these are.

(Refer Slide Time 27:48)

BCJR Algorithm

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

$$\alpha_{l+1}(s) = \frac{p(s, r_{t < l+1})}{\sum_{s' \in \sigma_l} p(s', r_{t < l+1})} = \sum_{s' \in \sigma_l} \frac{p(s, r_l | s', r_{t < l}) p(s', r_{t < l})}{\sum_{c} p(a, b, c)}$$

$$= \sum_{s' \in \sigma_l} p(s, r_l | s') p(s', r_{t < l})$$

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

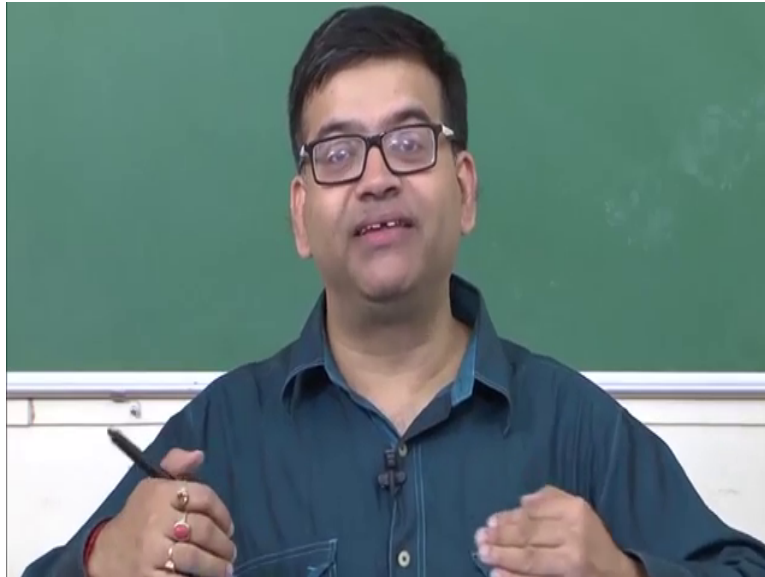
where σ_l is the set of all states at time l .

Handwritten notes and diagram:

- Left box: $\alpha_{2H}(0) = \gamma_x(0,0)\alpha_x(0) + \gamma_x(1,0)\alpha_x(1)$
- Right box: $\alpha_{2H}(j) = \gamma_x(0,j)\alpha_x(0) + \gamma_x(1,j)\alpha_x(1)$
- Diagram: A trellis with states 0 and 1 at time $t=l$ and $t=l+1$. Transitions are labeled with bit pairs: 0/01, 1/10, 0/00, 1/11.
- Additional notes: $\alpha_0(0) = ?$, $\alpha_0(1) = ?$

Now note initially we assume that the

(Refer Slide Time 27:52)



encoder is in all zero state. So if we assume the encoder is in all zero state then it is,

(Refer Slide Time 28:00)

BCJR Algorithm

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

$$\alpha_{l+1}(s) = \frac{p(s, r_{t < l+1})}{\sum_{s' \in \sigma_l} p(s', r_{t < l+1})} = \sum_{s' \in \sigma_l} \frac{p(s, r_l | s') p(s', r_{t < l})}{\sum_{c} p(a, b, c)}$$

$$\alpha_{2H}(0) = \gamma_x(0,0)\alpha_x(0) + \gamma_x(1,0)\alpha_x(1)$$

$$\alpha_{2H}(1) = \gamma_x(0,1)\alpha_x(0) + \gamma_x(1,1)\alpha_x(1)$$

$$= \sum_{s' \in \sigma_l} p(s, r_l | s') p(s', r_{t < l})$$

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

where σ_l is the set of all states at time l .

(12)

$\alpha_0(0) = ?$
 $\alpha_0(1) = ?$

it is going to stay in all zero state then in that case, we consider this probability as 1 and this all other possibility of it staying all other state is 0. So the initial value when we assume that the encoder is in all zero state we assume that alpha 0 at 0 is 1 and alpha 0 at at any other state is 0. So similarly we can

(Refer Slide Time 28:26)

BCJR Algorithm

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

$$\begin{aligned} \beta_l(s') &\equiv p(\mathbf{r}_{t>(l-1)}|s') && (13) \\ &= \sum_{s \in \sigma_{l+1}} p(\mathbf{r}_{t>(l-1)}, s|s') \\ &= \sum_{s \in \sigma_{l+1}} p(\mathbf{r}_{t>l}, \mathbf{r}_l, s|s') \\ &= \sum_{s \in \sigma_{l+1}} p(\mathbf{r}_{t>l}|s', s, \mathbf{r}_l) p(s, \mathbf{r}_l|s') \\ &= \sum_{s \in \sigma_{l+1}} p(\mathbf{r}_{t>l}|s) p(s, \mathbf{r}_l|s') \\ &= \sum_{s \in \sigma_{l+1}} \beta_{l+1}(s) \gamma_l(s', s) \end{aligned}$$

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

again we introduce a new variable s and sum over

(Refer Slide Time 28:46)

BCJR Algorithm

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

$$\begin{aligned} \beta_l(s') &\equiv p(\mathbf{r}_{t>(l-1)}|s') && (13) \\ &= \sum_{s \in \sigma_{l+1}} p(\mathbf{r}_{t>(l-1)}, s|s') \\ &= \sum_{s \in \sigma_{l+1}} p(\mathbf{r}_{t>l}, \mathbf{r}_l, s|s') \\ &= \sum_{s \in \sigma_{l+1}} p(\mathbf{r}_{t>l}|s', s, \mathbf{r}_l) p(s, \mathbf{r}_l|s') \\ &= \sum_{s \in \sigma_{l+1}} p(\mathbf{r}_{t>l}|s) p(s, \mathbf{r}_l|s') \\ &= \sum_{s \in \sigma_{l+1}} \beta_{l+1}(s) \gamma_l(s', s) \end{aligned}$$

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

all possible values of s, so then this becomes, from here we get this. Now

(Refer Slide Time 28:55)

BCJR Algorithm

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

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

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

we split this r into these 2 terms,

(Refer Slide Time 29:01)

BCJR Algorithm

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

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

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

so we get this expression.

(Refer Slide Time 29:05)

BCJR Algorithm

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

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

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

Now using Bayes rule I can separate out this term into 2 terms like this. And we know that, again let's go back to our Trellis section, so 0 0 1 1 1, this is

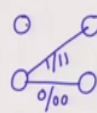
(Refer Slide Time 29:27)

BCJR Algorithm

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

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

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

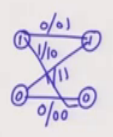


1 1 0 and this is 0 0 1. This is state 0, this is state 1. So if you are interested

(Refer Slide Time 29:37)

BCJR Algorithm

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

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


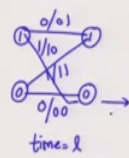
where σ_{l+1} is the set of all states at time $l + 1$.

in probability of r_t greater than l , that is this is your, this is your time l so probability of r_t

(Refer Slide Time 29:50)

BCJR Algorithm

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

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


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

greater than l given previous state s hat, next state s and r_l , it only depends on,

(Refer Slide Time 29:58)

BCJR Algorithm

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

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

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

so if you know the next state s you don't need information about the previous state, you don't need information about the current bit. So I can simplify this expression

(Refer Slide Time 30:12)

BCJR Algorithm

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

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

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

in this particular fashion and if we go back, this is nothing but our betas and this is our gamma. So let's compute beta for this particular code. So we are interested in computing beta 1 for 0 and beta 1 at state 1. So beta 1 at state 0 would be, so beta 1 at state 0, so we, so we are interested in computing beta at state 0, so this is sum over all those transitions which are ending at this state. So there are 2 transitions, one is this one, another is this one. So let's write the expression for this particular term. This we can write as beta at time $l + 1$, 1 times gamma l 0 1. So the contribution of this is

(Refer Slide Time 31:21)

BCJR Algorithm

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

$$\beta_l(s') \equiv \frac{p(r_{t>(l-1)}|s')}{\sum_{s \in \sigma_{l+1}} p(r_{t>(l-1)}, s | s')} \quad (13)$$

$$= \sum_{s \in \sigma_{l+1}} p(r_{t>l}, r_l, s | s')$$

$$= \sum_{s \in \sigma_{l+1}} p(r_{t>l} | s', s, r_l) p(s, r_l | s')$$

$$= \sum_{s \in \sigma_{l+1}} p(r_{t>l} | s) p(s, r_l | s')$$

$$= \sum_{s \in \sigma_{l+1}} \beta_{l+1}(s) \gamma_l(s', s)$$

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

beta l plus 1 corresponding to state 1 multiplied by gamma of this Trellis section, gamma l when the initial state is 0 and the next state is 1. So this, this, this will contribute this term plus there is another transition which is this, this one right. So we can write contribution of this as beta l plus 1 zero times gamma l 0 0.

(Refer Slide Time 31:59)

BCJR Algorithm

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

$$\beta_l(s') \equiv \frac{p(r_{t>(l-1)}|s')}{\sum_{s \in \sigma_{l+1}} p(r_{t>(l-1)}, s | s')} \quad (13)$$

$$= \sum_{s \in \sigma_{l+1}} p(r_{t>l}, r_l, s | s')$$

$$= \sum_{s \in \sigma_{l+1}} p(r_{t>l} | s', s, r_l) p(s, r_l | s')$$

$$= \sum_{s \in \sigma_{l+1}} p(r_{t>l} | s) p(s, r_l | s')$$

$$= \sum_{s \in \sigma_{l+1}} \beta_{l+1}(s) \gamma_l(s', s)$$

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

So this is our expression of beta l for state 0. Similarly we can compute beta l for state 1. So what are the 2 transitions which are ending at this state? One is this one, other one is this one. So let's write down the expression for this one, this one. So this will be beta l plus 1, 0 times gamma l 1 0 this is this term and what about this particular term, this will be given by beta l plus 1 1 times gamma l 1 1.

(Refer Slide Time 32:55)

BCJR Algorithm

- Similarly expression or the probability $\beta_l(s')$ can be written as

$$\beta_l(s') \equiv \frac{p(\mathbf{r}_{t>(l-1)} | s')}{\sum_{s \in \sigma_{l+1}} p(\mathbf{r}_{t>(l-1)}, s | s')} \quad (13)$$

$$= \sum_{s \in \sigma_{l+1}} p(\mathbf{r}_{t>l}, \mathbf{r}_l, s | s')$$

$$= \sum_{s \in \sigma_{l+1}} p(\mathbf{r}_{t>l} | s', s) p(\mathbf{r}_l | s) p(s, \mathbf{r}_l | s')$$

$$= \sum_{s \in \sigma_{l+1}} p(\mathbf{r}_{t>l} | s) p(s, \mathbf{r}_l | s')$$

$$= \sum_{s \in \sigma_{l+1}} \beta_{l+1}(s) \gamma_l(s', s)$$

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

So this is our expression for betas. So as you can see similar to the expression for alphas now these betas

(Refer Slide Time 33:09)

BCJR Algorithm

- Similarly expression or the probability $\beta_l(s')$ can be written as

$$\beta_l(s') \equiv \frac{p(\mathbf{r}_{t>(l-1)} | s')}{\sum_{s \in \sigma_{l+1}} p(\mathbf{r}_{t>(l-1)}, s | s')} \quad (13)$$

$$= \sum_{s \in \sigma_{l+1}} p(\mathbf{r}_{t>l}, \mathbf{r}_l, s | s')$$

$$= \sum_{s \in \sigma_{l+1}} p(\mathbf{r}_{t>l} | s', s) p(\mathbf{r}_l | s) p(s, \mathbf{r}_l | s')$$

$$= \sum_{s \in \sigma_{l+1}} p(\mathbf{r}_{t>l} | s) p(s, \mathbf{r}_l | s')$$

$$= \sum_{s \in \sigma_{l+1}} \beta_{l+1}(s) \gamma_l(s', s)$$

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

can be computed using,

(Refer Slide Time 33:13)

BCJR Algorithm

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

$$\beta_l(s') \equiv \frac{p(r_{t>(l-1)}|s')}{\sum_{s \in \sigma_{l+1}} p(r_{t>(l-1)}|s, s')} \quad (13)$$

$$\beta_l(s') \equiv \frac{p(r_{t>l}, r_l, s | s')}{\sum_{s \in \sigma_{l+1}} p(r_{t>l}, r_l, s | s')} \quad (13)$$

$$\beta_l(s') \equiv \frac{p(r_{t>l} | s', s) p(r_l | s) p(s, r_l | s')}{\sum_{s \in \sigma_{l+1}} p(r_{t>l} | s) p(s, r_l | s')}$$

$$\beta_l(s') \equiv \frac{p(r_{t>l} | s) p(s, r_l | s')}{\sum_{s \in \sigma_{l+1}} \beta_{l+1}(s) \gamma_l(s', s)}$$

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

so alphas can be computed using

(Refer Slide Time 33:15)



forward recursion and similarly betas can be computed using backward recursion. So then 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 the encoder is terminated, that means if the encoder is brought back to all zero state in that case, beta at end of Trellis, at end of the time, let's call beta at time k

(Refer Slide Time 33:51)

BCJR Algorithm

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

$$\beta_l(s') \equiv \frac{p(r_{t>(l-1)}|s')}{\sum_{s \in \sigma_{l+1}} p(r_{t>(l-1)}|s, s')} \quad (13)$$

$$\beta_l(s') \equiv \frac{p(r_{t>l}, r_l, s|s')}{\sum_{s \in \sigma_{l+1}} p(r_{t>l}, r_l, s|s')} \quad (13)$$

$$\beta_l(s') \equiv \frac{p(r_{t>l}|s', s) p(r_l, s|s')}{\sum_{s \in \sigma_{l+1}} p(r_{t>l}|s) p(s, r_l|s')}$$

$$\beta_l(s') \equiv \frac{p(r_{t>l}|s) p(s, r_l|s')}{\sum_{s \in \sigma_{l+1}} \beta_{l+1}(s) \gamma_l(s', s)}$$

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

which is the end of the block, at state 0 will be 1 and for all other state

(Refer Slide Time 33:57)

BCJR Algorithm

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

$$\beta_l(s') \equiv \frac{p(r_{t>(l-1)}|s')}{\sum_{s \in \sigma_{l+1}} p(r_{t>(l-1)}|s, s')} \quad (13)$$

$$\beta_l(s') \equiv \frac{p(r_{t>l}, r_l, s|s')}{\sum_{s \in \sigma_{l+1}} p(r_{t>l}, r_l, s|s')} \quad (13)$$

$$\beta_l(s') \equiv \frac{p(r_{t>l}|s', s) p(r_l, s|s')}{\sum_{s \in \sigma_{l+1}} p(r_{t>l}|s) p(s, r_l|s')}$$

$$\beta_l(s') \equiv \frac{p(r_{t>l}|s) p(s, r_l|s')}{\sum_{s \in \sigma_{l+1}} \beta_{l+1}(s) \gamma_l(s', s)}$$

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

it will be, in this case there are only 2 states, so for all other states it will be 0.

(Refer Slide Time 34:04)

BCJR Algorithm

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

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

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

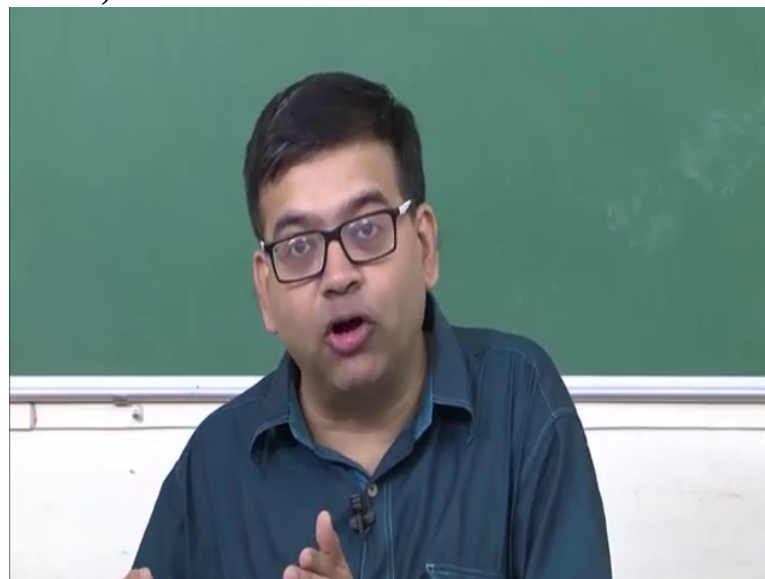
Handwritten notes on the slide:

- $\beta_x(0) = \beta_{RH}(0) \gamma_x(0,1) + \beta_{RH}(0) \gamma_x(0,0)$
- $\beta_x(1) = \beta_{RH}(0) \gamma_x(1,0) + \beta_{RH}(1) \gamma_x(1,1)$

Diagram on the right shows a trellis for time l with states 0 and 1 at time l and 0 and 1 at time $l+1$. Transitions are labeled with $0/0$ and $1/1$. A path is highlighted with red and green lines.

This is for the case when the convolutional encoder is brought back to all zero

(Refer Slide Time 34:11)



state, it is terminated. In case the convolutional encoder is not terminated, then we don't know in which state it has uh ended up with. So what we will do is in that case we will assume

(Refer Slide Time 34:38)

BCJR Algorithm

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

$$\beta_l(s') \equiv \frac{p(r_{t>(l-1)}|s')}{\sum_{s \in \sigma_{l+1}} p(r_{t>(l-1)}|s, s')}$$

$$\beta_l(s') \equiv \frac{p(r_{t>l}, r_l, s|s')}{\sum_{s \in \sigma_{l+1}} p(r_{t>l}|s', s, r_l) p(s, r_l|s')}$$

$$\beta_l(s') \equiv \frac{p(r_{t>l}|s) p(s, r_l|s')}{\sum_{s \in \sigma_{l+1}} p(r_{t>l}|s) p(s, r_l|s')}$$

$$\beta_l(s') \equiv \sum_{s \in \sigma_{l+1}} \beta_{l+1}(s) \gamma_l(s', s)$$

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

that it is equally likely to end up at all zero state or any other state. 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, we would assume that beta k 0 is half and beta k 1 is half. So this is for the case when convolutional encoder is not

(Refer Slide Time 34:56)

BCJR Algorithm

- Similarly expression or the probability $\beta_l(s')$ can be written as $\beta_k(0) = 1/2$
 $\beta_k(1) = 0/2$ (13)

$$\beta_l(s') \equiv \frac{p(r_{t>(l-1)}|s')}{\sum_{s \in \sigma_{l+1}} p(r_{t>(l-1)}|s, s')}$$

$$\beta_l(s') \equiv \frac{p(r_{t>l}, r_l, s|s')}{\sum_{s \in \sigma_{l+1}} p(r_{t>l}|s', s, r_l) p(s, r_l|s')}$$

$$\beta_l(s') \equiv \frac{p(r_{t>l}|s) p(s, r_l|s')}{\sum_{s \in \sigma_{l+1}} p(r_{t>l}|s) p(s, r_l|s')}$$

$$\beta_l(s') \equiv \sum_{s \in \sigma_{l+1}} \beta_{l+1}(s) \gamma_l(s', s)$$

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

terminated. That means 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 35:07)

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 metric gamma. Now from definition gammas can be written like this. So this can be written as joint probabilities of being in previous state s prime next state s given a received sequence at time l r l divided by probability of being in previous state s prime. Now this I introduce the term this so I

(Refer Slide Time 35:40)

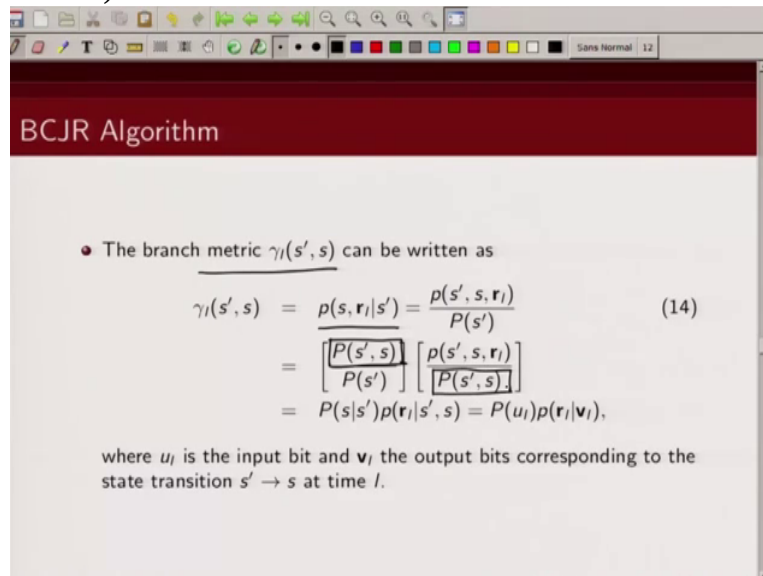
The branch metric $\gamma_l(s', s)$ can be written as

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

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 .

add this term in the numerator, similarly I add this term in the denominator, Ok.

(Refer Slide Time 35:46)

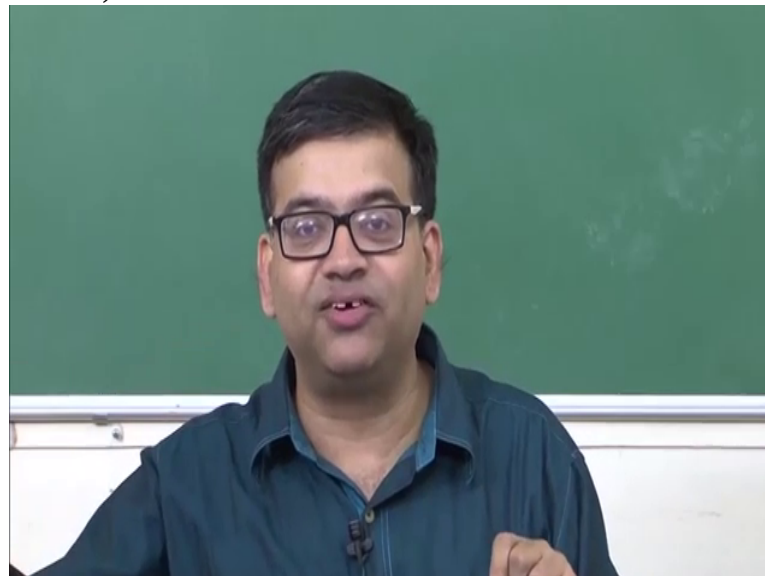


The screenshot shows a presentation slide with a red header titled "BCJR Algorithm". Below the header, there is a bullet point: "• The branch metric $\gamma_l(s', s)$ can be written as". This is followed by a mathematical derivation for equation (14):
$$\begin{aligned}\gamma_l(s', s) &= \frac{p(s, \mathbf{r}_l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

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 this, this quantity can be written as probability of s given s prime and this probability can be written as probability of \mathbf{r}_l given previous state s prime and next state s which can also be written as probability of \mathbf{r}_l given transmitted sequence \mathbf{v}_l multiplied by a priori probability of getting u_l . So note that this probability will be 1 only when there is a valid transition from

(Refer Slide Time 36:22)



state s prime to s , otherwise this will be 0, Ok.

(Refer Slide Time 36:30)

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

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

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 .

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 36:46)

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 .

Now if we consider an additive white gaussian noise channel, we can write this probability of \mathbf{r}_l given \mathbf{v}_l in this particular fashion. So gamma for an a w g n channel will then be given by this expression. So note this depends on a priori

(Refer Slide Time 37:09)

BCJR Algorithm

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

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

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 .

probability of u_l . It depends on the Euclidean distance between r_l and v_l . Now let us assume that we are considering a binary phase shift keying. So in other words basically we have bits mapped to plus 1 and minus 1 let's say or plus

(Refer Slide Time 37:31)

BCJR Algorithm

BPSK

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

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

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 .

E_s and minus E_s , Ok. 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

(Refer Slide Time 37:49)

BPSK

- 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 .

signal to noise ratio. And if you look at this particular term, so here there is a

(Refer Slide Time 37:56)

BPSK

- 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 .

r l square term, there is a v l square term and then there is minus 2 r l v l term. So this r l,

(Refer Slide Time 38:09)

BCJR Algorithm

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 .

Handwritten notes on the slide:
- A red note above the exponential term: $r_l^2 + v_l^2 - 2r_l v_l$
- A blue box around $P(u_l)$
- A red box around $\left(\sqrt{\frac{E_s}{\pi N_0}} \right)^n$
- A red box around $e^{-\frac{E_s}{N_0} \|r_l - v_l\|^2}$

r_l square term that does not depend on what v_l is. And since we are considering a BPSK modulated signal, so v_l whether u_l is

(Refer Slide Time 38:23)



minus 1 or plus 1, this will basically be the same. This will be just 1.

(Refer Slide Time 38:28)

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 .

$r_l^2 + v_l^2 - 2r_l v_l$

So the only term that is changing with choice of v_l is this particular term. So what we can simplify this γ_l , we can just simplify our γ_l like this. So this is basically probability of u_l and exponential minus E_s by $n N_0$ and this is basically two times $r_l \cdot v_l$ so it becomes dot product between the received sequence and uh this transmitted codeword v_l . So, and of course there is some constant, there is some constant term k_1

(Refer Slide Time 39:12)

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 .

$\gamma_l(s, s) = P(u_l) e^{-\frac{E_s}{2 N_0} r_l v_l}$

which is common. So in nutshell then, our γ_l depends on this term, right and it depends on what the initial a priori probability of u_l is. So for an additive white gaussian noise channel when we are applying b p s k modulation then we can simplify our expression for γ_l . So this can be written as E_s raised to power minus E_s by N_0 by 2 times $r_l \cdot v_l$.

(Refer Slide Time 39:49)

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 .

Now what is this $r_l \cdot v_l$? We will illustrate this with an example when we solve, when we show an example Ok. So the point which I am trying to make is that this expression that you see for computation of gamma for additive white gaussian noise, it essentially depends on two terms. One is this, and another is this term.

Next

(Refer Slide Time 40:16)

BCJR Algorithm

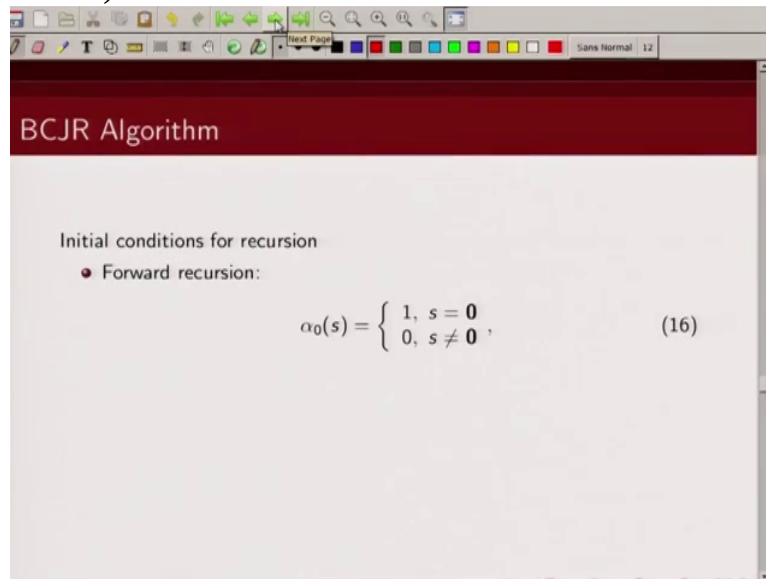
- For a continuous output AWGN channel, if $s' \rightarrow s$ is a valid state transition,

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

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 .

- 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.

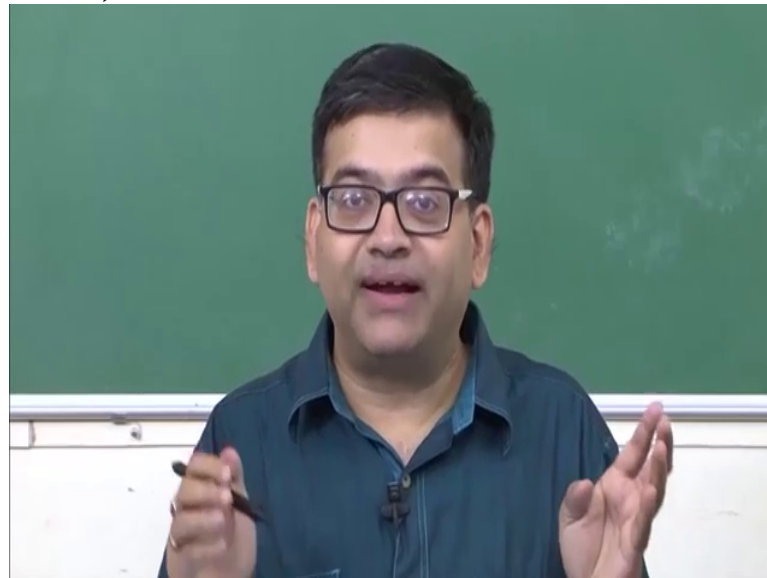
(Refer Slide Time 40:17)



The image shows a screenshot of a presentation slide titled "BCJR Algorithm". The slide content includes the text "Initial conditions for recursion" followed by a bullet point "• Forward recursion:". Below this, the initial condition for the forward recursion is given as a piecewise function:
$$\alpha_0(s) = \begin{cases} 1, & s = 0 \\ 0, & s \neq 0 \end{cases} \quad (16)$$

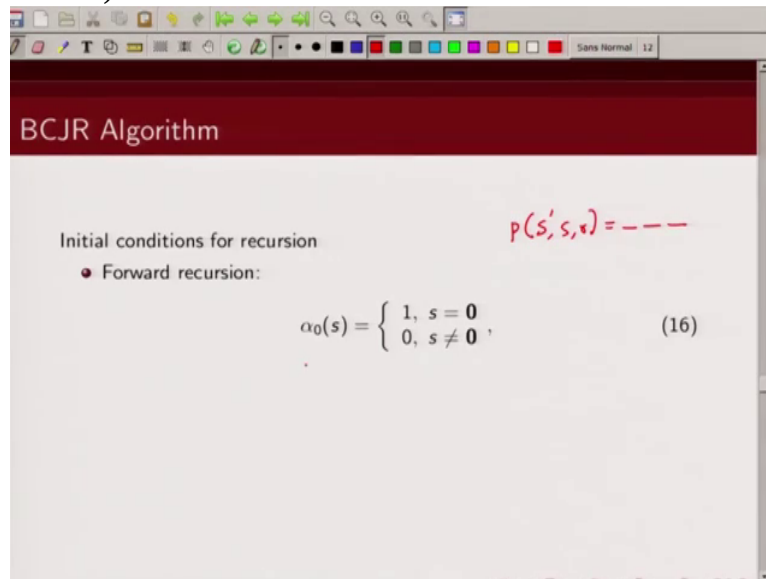
I have already specified now that

(Refer Slide Time 40:22)



our joint probability of, the joint probability that we computed, it's 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

(Refer Slide Time 40:38)



BCJR Algorithm

Initial conditions for recursion

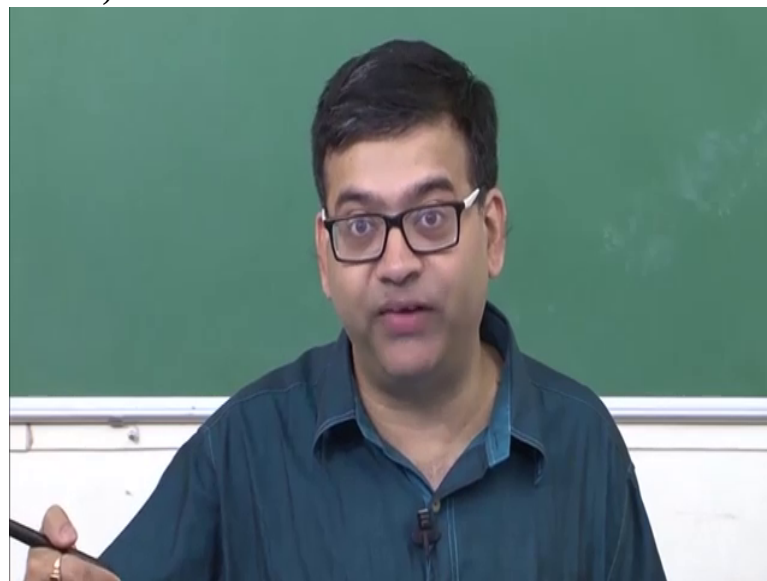
- Forward recursion:

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

$p(s', s, r) = \text{---}$

usually our encoder is in all zero state to start off with

(Refer Slide Time 40:43)



and that's why we assume that

(Refer Slide Time 40:46)

BCJR Algorithm

Initial conditions for recursion

- Forward recursion:

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

p(s', s, r) = ----

alpha times 0 is 1 for the state 0 and it is 0 for all other states. Similarly

(Refer Slide Time 40:59)

BCJR Algorithm

Initial conditions for recursion

- Forward recursion:

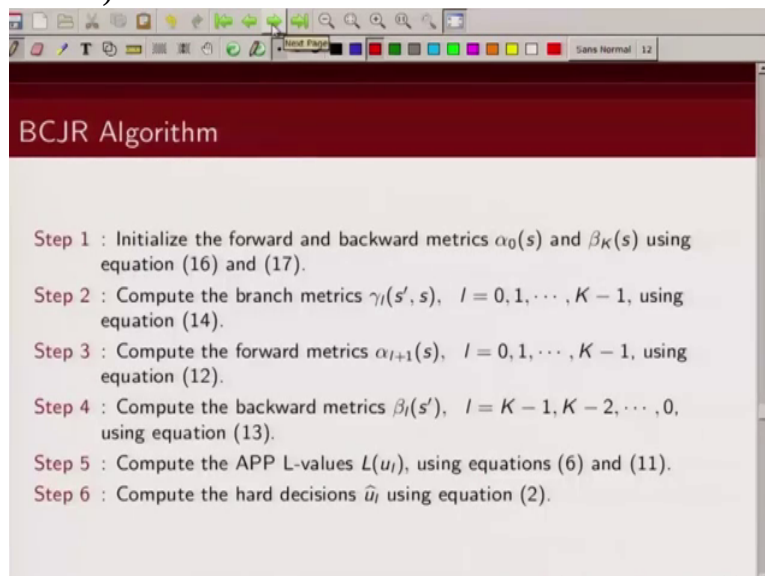
$$\alpha_0(s) = \begin{cases} 1, & s = 0 \\ 0, & s \neq 0 \end{cases} \quad (16)$$

- Backward recursion:

$$\beta_{\kappa}(s) = \begin{cases} 1, & s = 0 \\ 0, & s \neq 0 \end{cases} \quad (17)$$

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 our block which is our k, beta k will be 1 for state 0 and 0 for all other states. So these are our initial conditions for computing the recursion for, for forward recursion as well as backward recursion. So now then,

(Refer Slide Time 41:31)

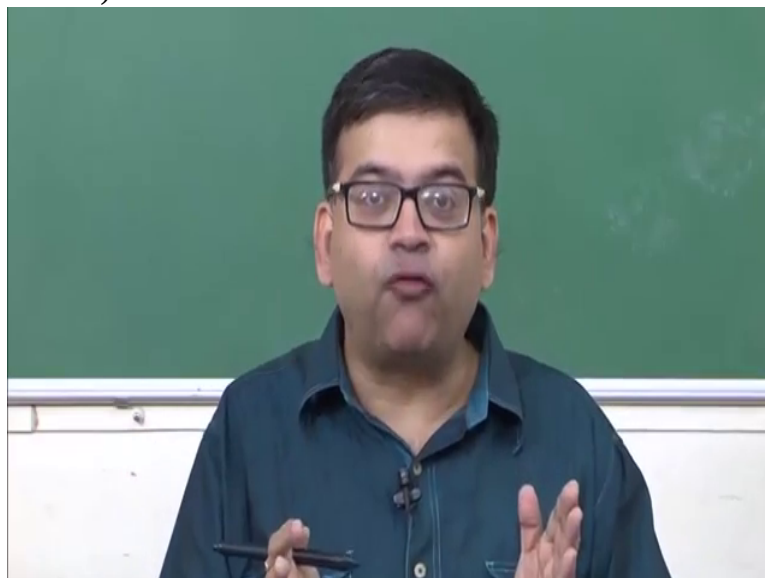


The image shows a presentation slide titled "BCJR Algorithm" with a red header. The slide lists six steps in a numbered list:

- 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).

to recap how do we compute the a posteriori probability. The first thing is we need to initialize the values of alpha times 0 and beta times 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 this branch metric gamma. So the first thing I need to do is I need to compute this branch metric gamma. So I will compute this branch metric for all

(Refer Slide Time 42:06)



valid transitions and for all time instances. So that's the second step.

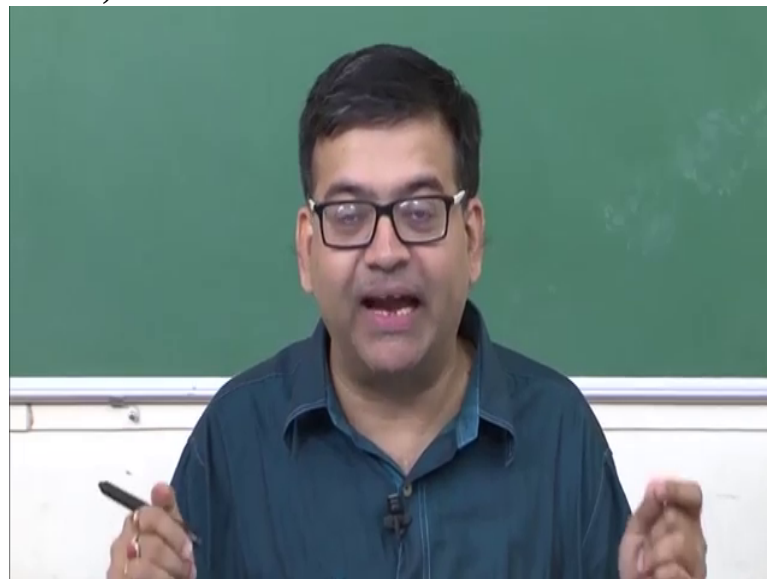
(Refer Slide Time 42:13)

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 $L(u_l)$, using equations (6) and (11).
- Step 6 : Compute the hard decisions \hat{u}_l using equation (2).

The third step is once I compute this branch metric 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

(Refer Slide Time 42:34)



then I can compute the a posteriori probability because I

(Refer Slide Time 42:40)



have shown that it is a basically product of these three terms. So I can then compute

(Refer Slide Time 42:46)

A screenshot of a presentation slide titled "BCJR Algorithm". The slide is displayed in a window with a standard operating system interface. The title is in a dark red header. The content consists of six numbered steps, each describing a part of the algorithm. The text is in a serif font, and some terms are underlined in red. The window's title bar and toolbar are visible at the top.

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 $L(u_l)$, using equations (6) and (11).
- Step 6** : Compute the hard decisions \hat{u}_l using equation (2).

these A P P values and once I have these A P P

(Refer Slide Time 42:52)

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).

value I will take a hard decision based on whether this is greater than 0 or plus 1. So the final thing that I am going to do is I am going to take a hard decision based on what is the value of this A P P value, Ok.

So let's now

(Refer Slide Time 43:14)

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

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

show the same using an example. So we are going to consider an example to illustrate how we can do b c j r decoding. So we are considering a rate 1 by 2 convolutional code with memory 1 whose generator sequence is basically given by this. The generator matrix is given by this. We are considering b p s k modulation and we are assuming that initial probability u l is equally likely to be plus 1

(Refer Slide Time 43:49)

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 BPSK, $P(u)$

$$\mathbf{r} = (\mathbf{r}_0, \mathbf{r}_1, \mathbf{r}_2, \mathbf{r}_3) = (r_0^{(0)}, r_0^{(1)}, r_1^{(0)}, r_1^{(1)}, r_2^{(0)}, r_2^{(1)}, r_3^{(0)}, r_3^{(1)})$$

$$= (+0.8, +0.1; +1.0, -0.5; -1.8, +1.1; +1.6, -1.6).$$

or minus 1. So we are assuming it is equally likely to be, it is plus 1 with probability half and minus 1 with probability half. We are

(Refer Slide Time 43:58)

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 BPSK, $P(u)$

$$\mathbf{r} = (\mathbf{r}_0, \mathbf{r}_1, \mathbf{r}_2, \mathbf{r}_3) = (r_0^{(0)}, r_0^{(1)}, r_1^{(0)}, r_1^{(1)}, r_2^{(0)}, r_2^{(1)}, r_3^{(0)}, r_3^{(1)})$$

$$= (+0.8, +0.1; +1.0, -0.5; -1.8, +1.1; +1.6, -1.6).$$

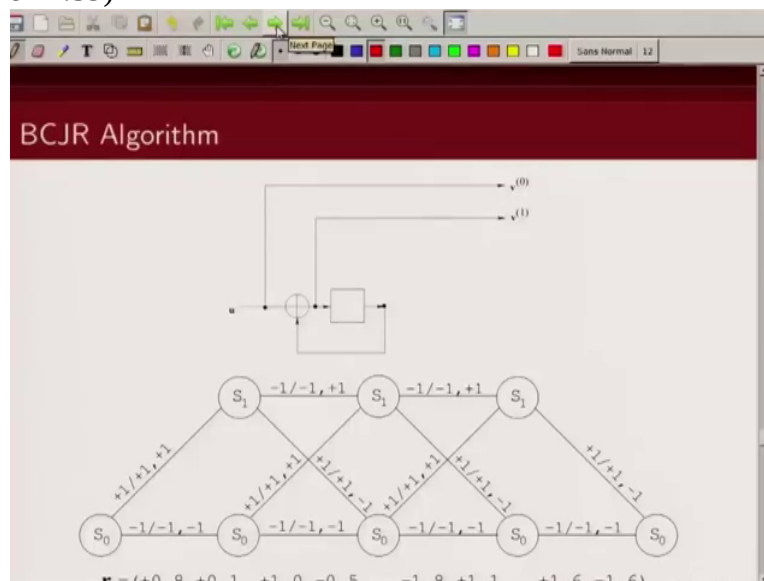
considering an a w g n channel with s n r of 1 by 4 and we are assuming that recieved signal are normalized by under root E of s. So what we are receiving is this particular sequence. The question I am interested is if the recieved sequence is this, I am interested in estimating what was my information

(Refer Slide Time 44:24)



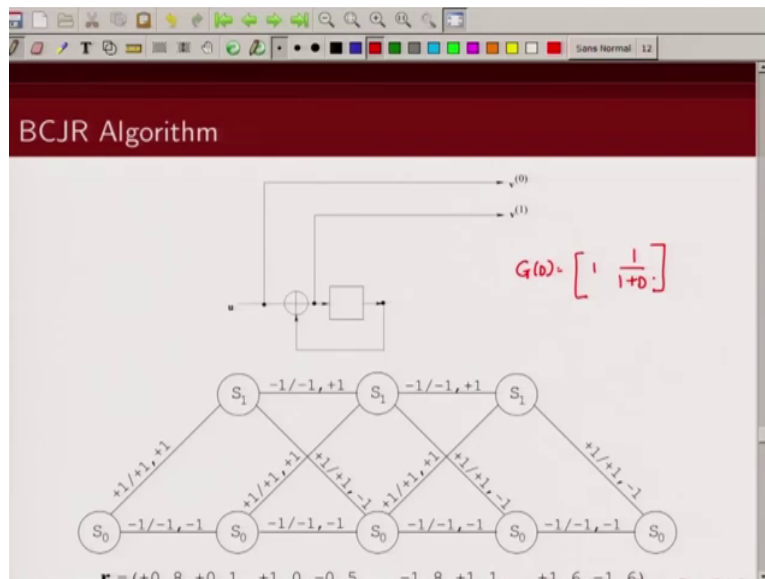
sequence. So to solve this problem what we need to do is we need to compute the a posteriori L value. Now to compute the a posteriori L value, we will first have to compute alpha, beta and gammas Ok and eventually we will compute the a posteriori L value and then we will take a hard decision on that to decide, estimate our information sequence. So this is the

(Refer Slide Time 44:59)



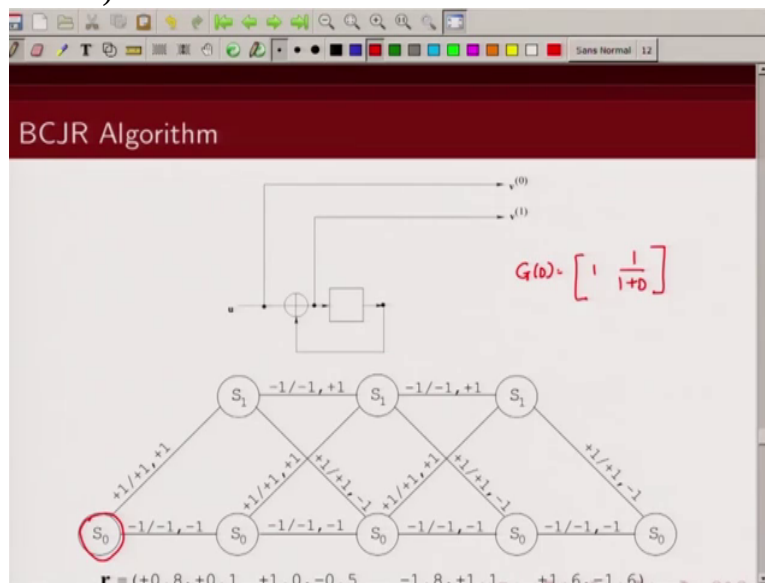
convolutional encoder that we have considered. This is basically G of D is rate 1 by 2 code and its

(Refer Slide Time 45:10)



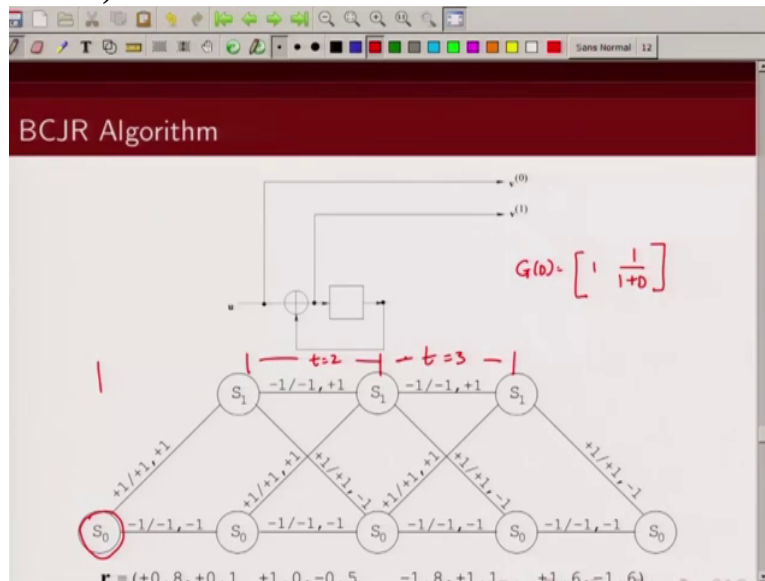
corresponding Trellis diagram is this. For simplicity I just considered 4 time instances. So initially I assume encoder is in all zero state

(Refer Slide Time 45:21)



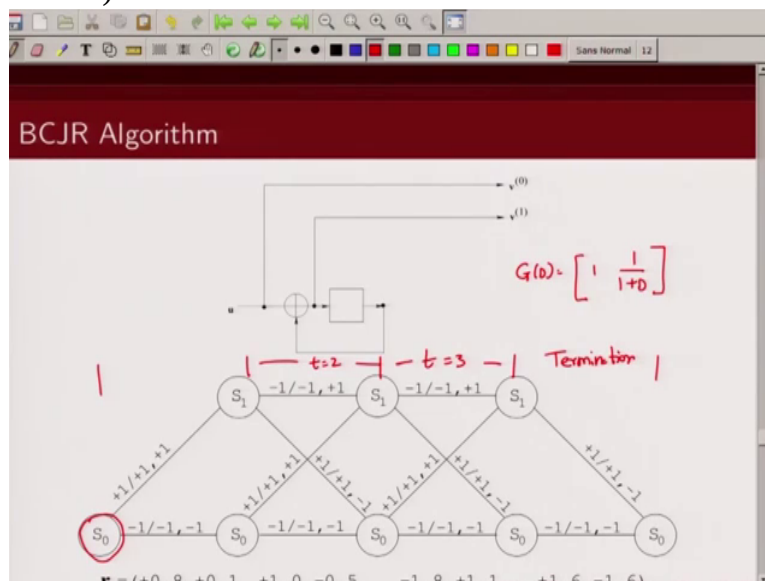
which is denoted by S_0 and it gets some bits. It moves to either S_0 or S_1 depending on what bits it received. This is first time instance; this is t equal to 2. This is, this is t equal to 3. And then after this what

(Refer Slide Time 45:42)



I am doing is I am terminating this encoder back to all zero state. So this is termination phase. So I bring this encoder

(Refer Slide Time 45:53)



back to all zero state. Now this is a rate 1 by 2 codes, for each Trellis section I am receiving 2 bits. So at time t equal to 1, what I recieved is these 2 bits, point 8 and plus point 1. For t equal to 2, I recieved these 2, plus point 1 and minus point 5. For t equal to 3, I recieved minus 1 point 8 and plus 1 point 1 and for, during the termination phase I recieved plus 1 point 6 and minus 1 point 6. Please note I am interested in, given this recieved sequence I am interested in estimating what was the information bit that was transmitted at time t equal to 1. What was the information bit that was transmitted at time t equal to 2. What was the information bit that was transmitted at time t equal to 3? So

(Refer Slide Time 46:56)

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:

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

as we said the first step was initializing alphas and betas for recursion; so since we started with all zero state, alpha at time zero for state 0 is 1, and for other states, which is state 1 it is zero. And since we are terminating

(Refer Slide Time 47:22)

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:

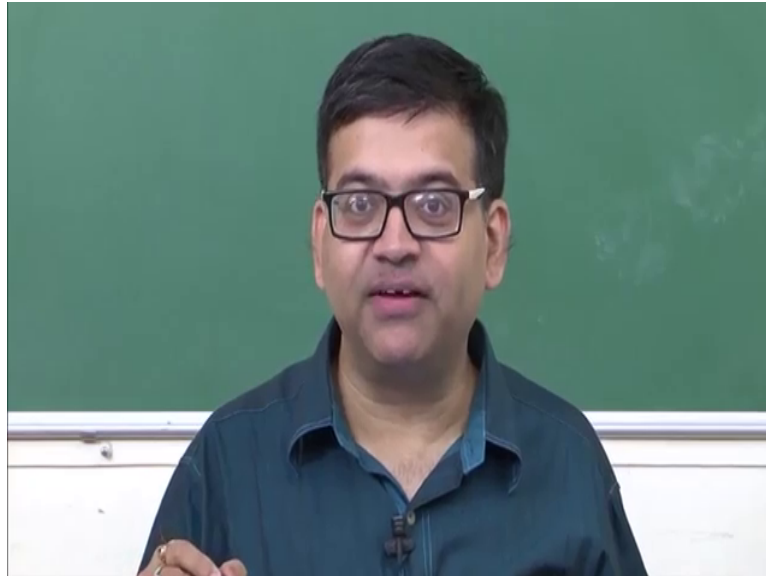
$$\alpha_0(s) = \begin{cases} 1, & s = 0 \\ 0, & s \neq 0 \end{cases}$$

- Backward recursion:

$$\beta_K(s) = \begin{cases} 1, & s = 0 \\ 0, & s \neq 0 \end{cases}$$

this encoder, so beta k times t equal to 4 is 1 for state 0 and it is 0 for other states which is state 1. So that's the first step. Initializing the forward and backward metric for time t equal to 0 and time t at end of the block, in our example t equal to 4. So once we have initialized our

(Refer Slide Time 47:52)



alphas and betas next we need to compute alphas and betas for other time instances and for that we would

(Refer Slide Time 48:04)

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

need 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, say call it k times probability initial a priori probability u_l and we have exponential plus E_s by N naught 2 times r_l dot v_l . Now in this particular example

(Refer Slide Time 48:33)

BCJR Algorithm: Example

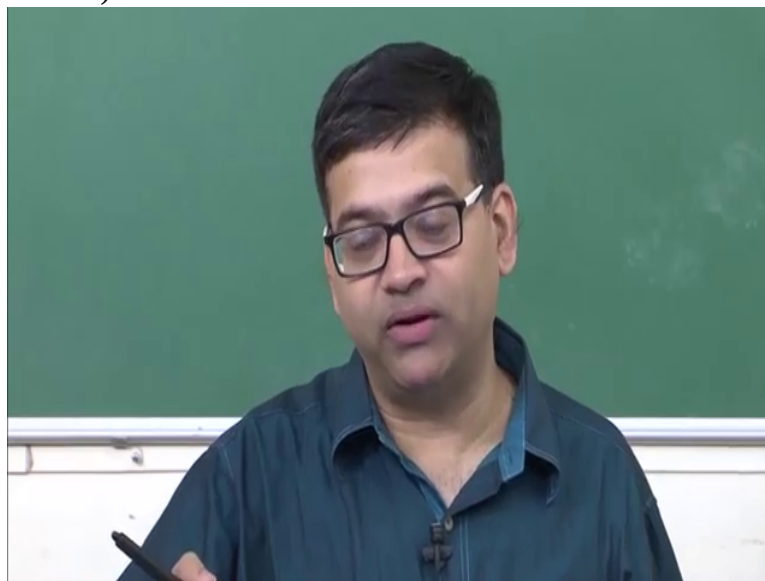
Step 2 : Compute the branch metrics $\gamma_l(s', s)$, $l = 0, 1, \dots, K - 1$, using equation (14).

$\gamma_l(s', s) = K_l P(u_l) e^{+b_{l,s} 2(x_l - u_l)}$

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

we are assuming that a priori it is equally likely to be plus one or minus one. So this

(Refer Slide Time 48:43)



probability will be half whether u_l is plus 1 or minus 1. So we can just, this will be

(Refer Slide Time 48:49)

BCJR Algorithm: Example

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

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

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

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 exponential plus E_s by N naught 2 times r_l dot v_l . Now in our example E_s by N naught is 1 by 4.

(Refer Slide Time 49:14)

BCJR Algorithm: Example

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

$\gamma_l(s', s) = K_l P(u_l) e^{\frac{E_s}{N_0} 2(r_l \cdot v_l)}$
 $= K_2 e^{\frac{E_s}{N_0} 2(r_l \cdot v_l)}$

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

Just go back, E_s by N naught is 1 by 4

(Refer Slide Time 49:19)

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

$$\mathbf{r} = (r_0, r_1, r_2, r_3) = (r_0^{(0)}, r_0^{(1)}; r_1^{(0)}, r_1^{(1)}; r_2^{(0)}, r_2^{(1)}; r_3^{(0)}, r_3^{(1)})$$

$$= (+0.8, +0.1; +1.0, -0.5; -1.8, +1.1; +1.6, -1.6).$$

BPSK, $P(u) \begin{cases} +1/2 \\ -1/2 \end{cases}$

4. So then we can write this as,

(Refer Slide Time 49:24)

BCJR Algorithm: Example

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

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

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

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

just ignore the constant term. I can write this as $r_l \dot{v}_l$ by 2. Now how do we compute $r_l \dot{v}_l$? Let's take an example.

(Refer Slide Time 49:38)

BCJR Algorithm: Example

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

$$\gamma_x(s', s) = K_1 P(u_x) e^{\frac{Es}{N_0} 2(x_s - v_x)}$$

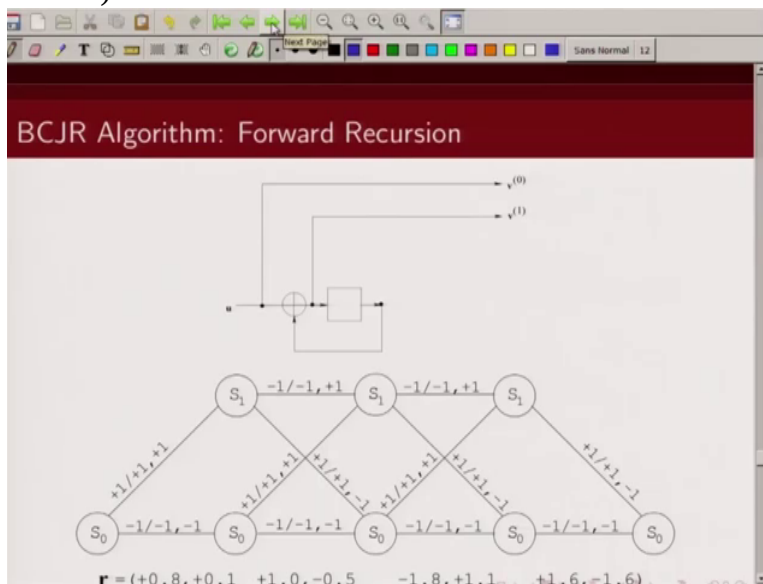
$$= K_2 e^{\frac{Es}{N_0} 2(x_s - v_x)}$$

$$= e^{\frac{r_s - v_s}{2}}$$

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

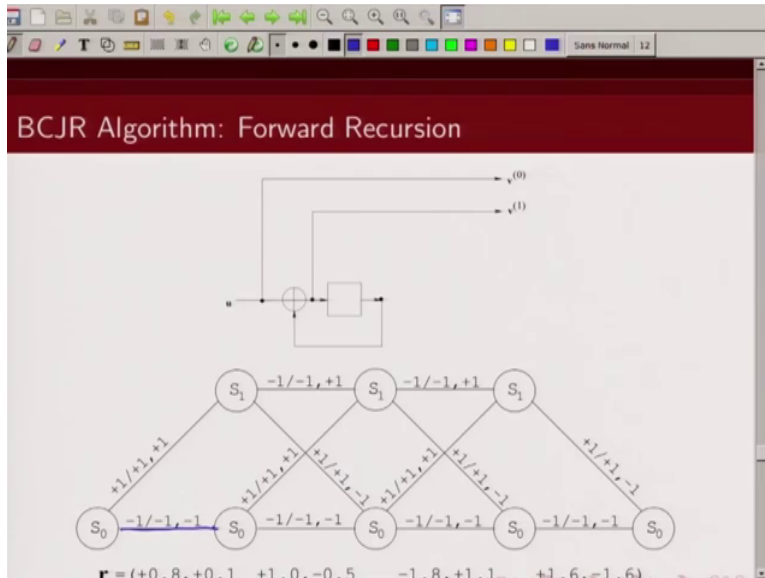
Let's take this. gamma at time t equal to 0, when the initial state is S 0 and final state is S 0, so what is this? This corresponds

(Refer Slide Time 49:52)

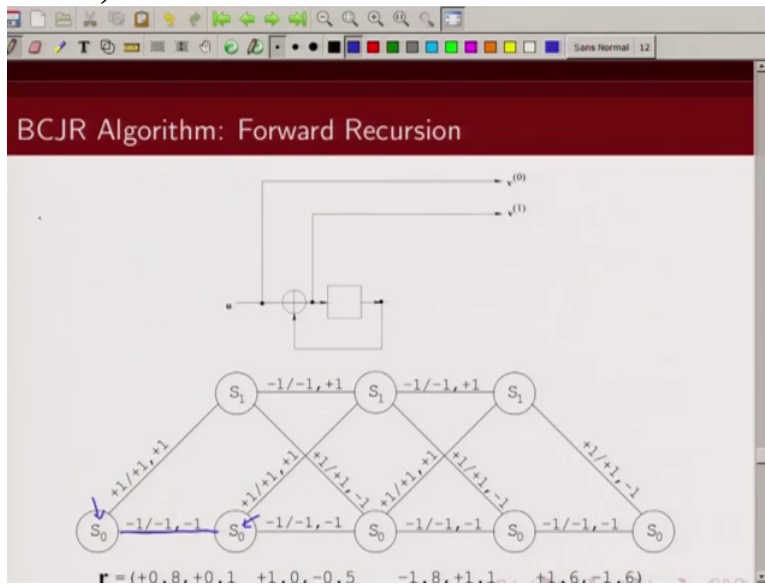


to branch metric for

(Refer Slide Time 49:55)

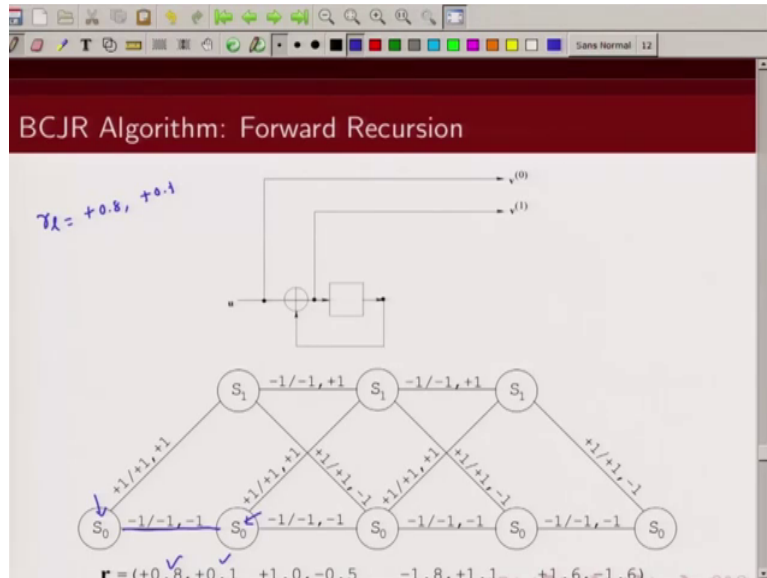


this path. This is time t equal to 0. Initial state is s_0 , final state is s_0 . Now what is (Refer Slide Time 50:03)



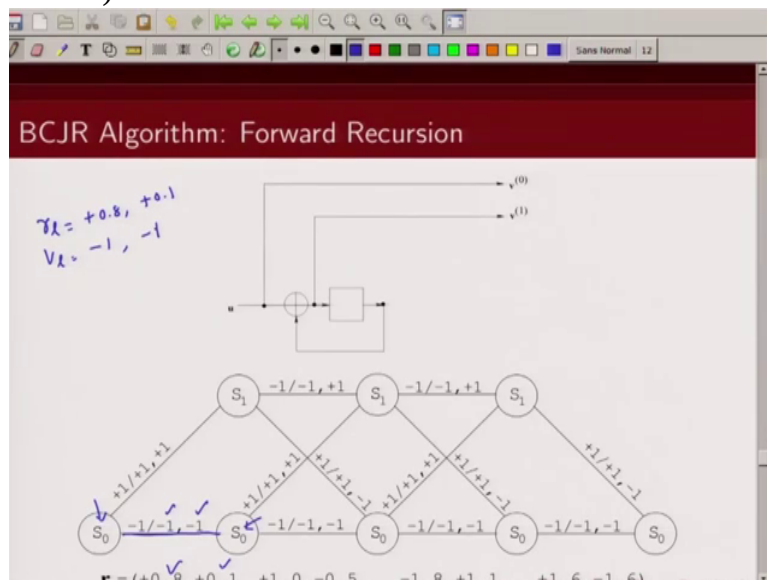
r L, what is the received sequence corresponding to this Trellis section? The received sequence is given by this. This is plus point 8 and plus point 1.

(Refer Slide Time 50:15)



Now what is the v_l corresponding to this transition? This is minus 1 and minus 1. So v_l is minus 1 and minus

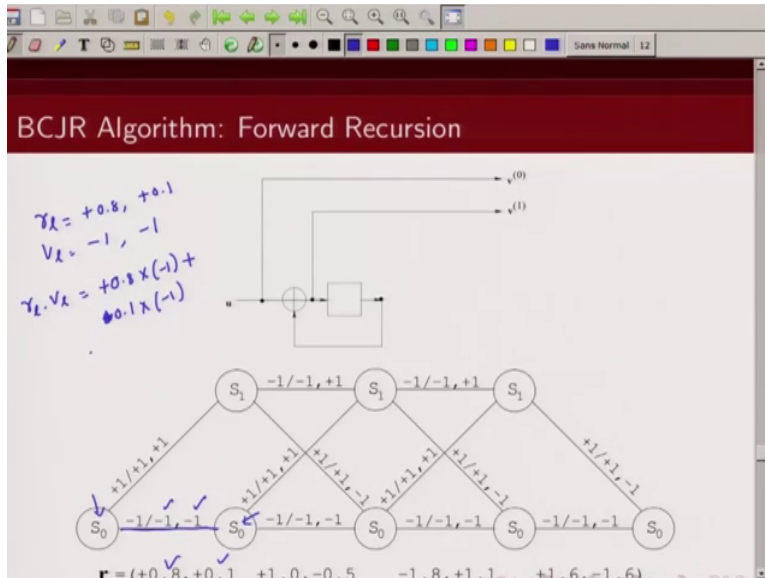
(Refer Slide Time 50:26)



1. Then this dot product can be written as plus point 8 into minus 1 plus point 1 into minus 1.

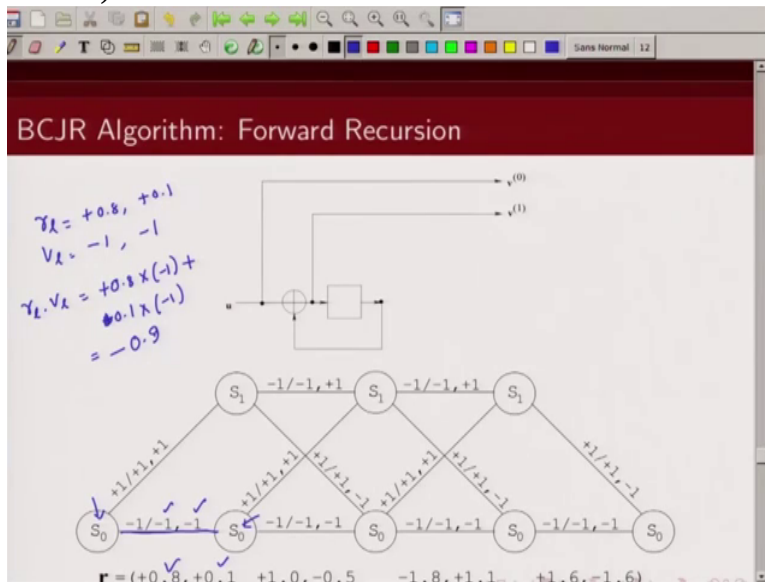
So

(Refer Slide Time 50:43)



this will be minus point 9. So $r \cdot v$ is minus point 9.

(Refer Slide Time 50:50)



So if we plug that in here, minus point 9 by

(Refer Slide Time 50:54)

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 notes on the right side of the slide:

$$\gamma_x(s', s) = K_1 P(u_x) e^{\frac{E_s}{N_0} 2(x_s - v_x)}$$

$$= K_2 e^{\frac{E_s}{N_0} 2(x_s - v_x)}$$

$$= e^{\frac{r_x - v_x}{2}}$$

2, so this is minus point 4 5 which is given by this. Now you can take any other example.

Let's just take this example. gamma 1 is S1 S 0, so what is this, so gamma 1

(Refer Slide Time 51:11)

BCJR Algorithm: Forward Recursion

Handwritten notes on the left side of the slide:

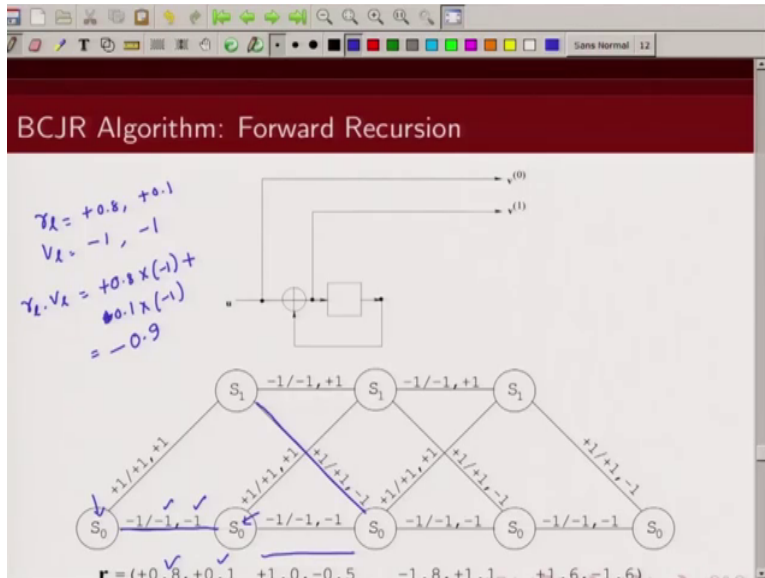
$$r_x = +0.8, +0.1$$

$$v_x = -1, -1$$

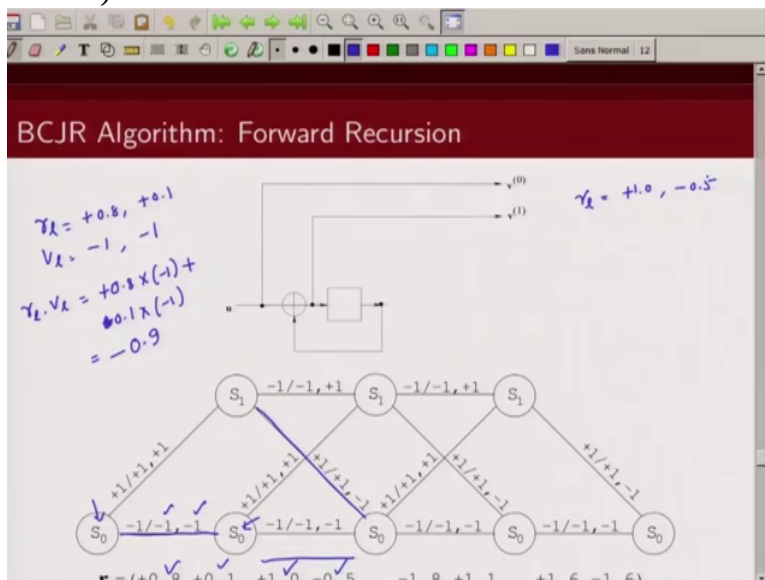
$$r_x \cdot v_x = +0.8 \times (-1) + 0.1 \times (-1) = -0.9$$

is time instance t equal to 1. so we were talking about this. And initial state is S 1; final state is S 0 so we are talking about this transition.

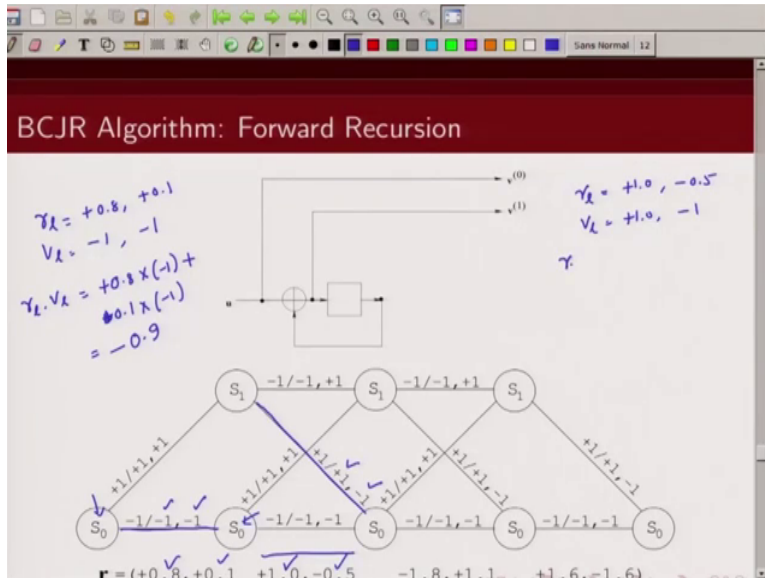
(Refer Slide Time 51:24)



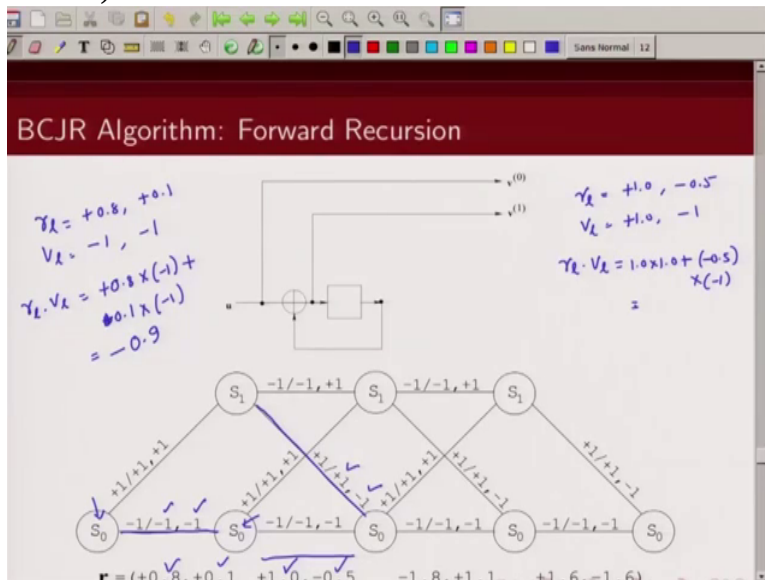
Now what is r_l corresponding to this transition? It is plus 1 and minus point 5. So this is plus 1 and minus point 5. What is v_l corresponding to
(Refer Slide Time 51:39)



this transition? It is given by plus 1 and minus 1, so v_l is plus 1 and minus 1; So then what would be r_l
(Refer Slide Time 51:52)



dot v l in this example? It is 1 into 1 plus minus point 5 into minus 1. So this will be (Refer Slide Time 52:04)



1 point 5. If you plug that in here, 1 point 5 by

(Refer Slide Time 52:10)

BCJR Algorithm: Example

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

$\gamma_0(S_0, S_0)$	$= e^{-0.45} = 0.6376$	$\gamma_l(s', s) = K_l P(U_l) e^{\frac{Es}{N_0} 2(x_l \cdot v_l)}$ $= K_l e^{\frac{Es}{N_0} 2(x_l \cdot v_l)}$ $= e^{\frac{r_{l, v_l}}{2}}$
$\gamma_0(S_0, S_1)$	$= e^{0.45} = 1.5683$	
$\gamma_1(S_0, S_0)$	$= e^{-0.25} = 0.7788$	
$\gamma_1(S_0, S_1)$	$= e^{0.25} = 1.2840$	
$\gamma_1(S_1, S_1)$	$= e^{-0.75} = 0.4724$	
$\gamma_1(S_1, S_0)$	$= e^{0.75} = 2.1170$	
$\gamma_2(S_0, S_0)$	$= e^{0.35} = 1.4191$	
$\gamma_2(S_0, S_1)$	$= e^{-0.35} = 0.7047$	
$\gamma_2(S_1, S_1)$	$= e^{1.45} = 4.2631$	
$\gamma_2(S_1, S_0)$	$= e^{-1.45} = 0.2346$	
$\gamma_3(S_0, S_0)$	$= e^0 = 1.0$	
$\gamma_3(S_1, S_0)$	$= e^{1.6} = 4.9530$	

2, so this would be e raised to power point 7 5. And that's what it is, Ok. So I hope it's clear how we can compute the branch metric.

(Refer Slide Time 52:22)



Please note we have ignored the common term

(Refer Slide Time 52:25)

BCJR Algorithm: Example

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

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

Handwritten notes:
 $\gamma_l(s', s) = K_2 P(u_l) e^{\frac{Es}{4} 2(x_l - v_l)}$
 $= K_2 e^{\frac{Es}{2} 2(x_l - v_l)}$
 $= e^{\frac{r_l \cdot v_l}{2}}$

which is common for computation of all of them. We are just computing the term which would be different based on what state transition that we are considering. So similarly we can compute gammas for other time instance t equal to 1, t equal to 2 and for all other valid state transitions. Now the next step is

(Refer Slide Time 52:49)

BCJR Algorithm: Forward Recursion

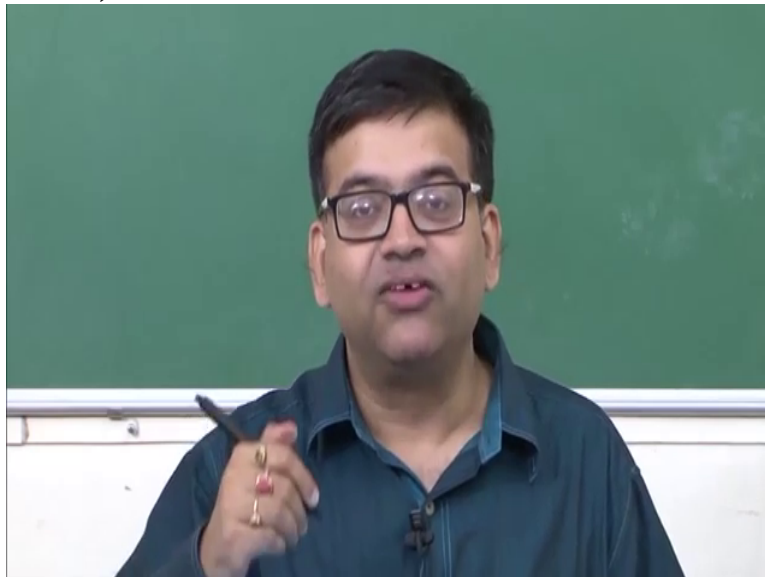
Handwritten notes on the left:
 $\gamma_l = +0.8, +0.1$
 $v_l = -1, -1$
 $\gamma_l \cdot v_l = +0.8 \times (-1) + 0.1 \times (-1) = -0.9$

Handwritten notes on the right:
 $\gamma_l = +1.0, -0.5$
 $v_l = +1.0, -1$
 $\gamma_l \cdot v_l = 1.0 \times 1.0 + (-0.5) \times (-1) = 1.5$

The trellis diagram shows states S_0 and S_1 at time $t=0, 1, 2$. Transitions are labeled with bit pairs (u, v) . For example, from S_0 to S_1 at $t=0$, the transition is $(+1, +1)$. The bottom row shows the branch metric values for each transition.

once we know our gamma we have already initialized our alphas and betas so the next

(Refer Slide Time 52:55)



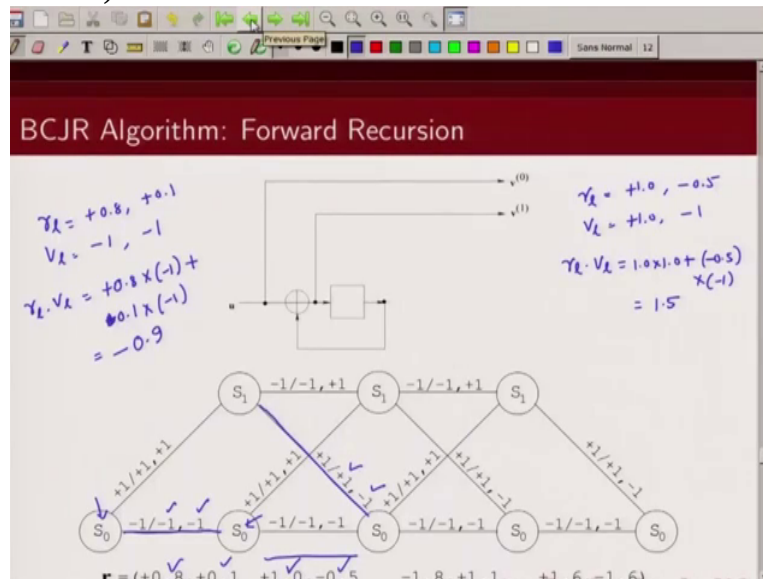
step would be to compute alphas and betas. And this is shown here. Now we have already illustrated how we can compute our alphas and betas. When we

(Refer Slide Time 53:11)

A screenshot of a presentation slide. The title is "BCJR Algorithm: Forward Recursion". Below the title, it says "Step 3 : Compute the forward metrics $\alpha_{l+1}(s)$, $l = 0, 1, \dots, K - 1$, using equation (12).". The slide lists several equations for the forward metrics:
$$\begin{aligned}\alpha_1(S_0) &= \alpha_0(S_0)\gamma_0(S_0, S_0) = 0.6376 \quad (0.2890) \\ \alpha_1(S_1) &= \alpha_0(S_0)\gamma_0(S_0, S_1) = 1.5683 \quad (0.7110) \\ \alpha_2(S_0) &= \alpha_1(S_0)\gamma_1(S_0, S_0) + \alpha_1(S_1)\gamma_1(S_1, S_0) = 3.8167 \quad (0.7099) \\ \alpha_2(S_1) &= \alpha_1(S_0)\gamma_1(S_0, S_1) + \alpha_1(S_1)\gamma_1(S_1, S_1) = 1.5595 \quad (0.2901) \\ \alpha_3(S_0) &= \alpha_2(S_0)\gamma_2(S_0, S_0) + \alpha_2(S_1)\gamma_2(S_1, S_0) = 5.7821 \quad (0.3824) \\ \alpha_3(S_1) &= \alpha_2(S_0)\gamma_2(S_0, S_1) + \alpha_2(S_1)\gamma_2(S_1, S_1) = 9.3379 \quad (0.6176)\end{aligned}$$

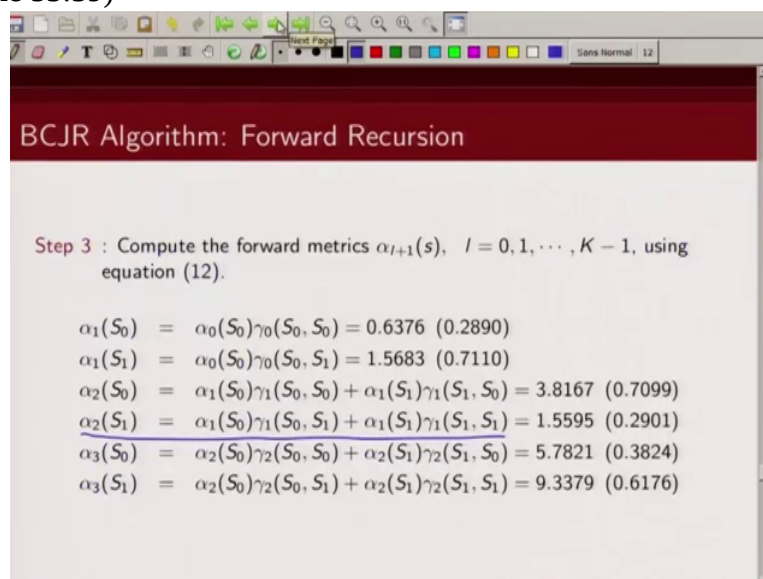
explained how to compute these forward recursion and backward recursion we can take another example. Let's just take this example. alpha 2 at state S 1, so alpha 2 will correspond to,

(Refer Slide Time 53:28)



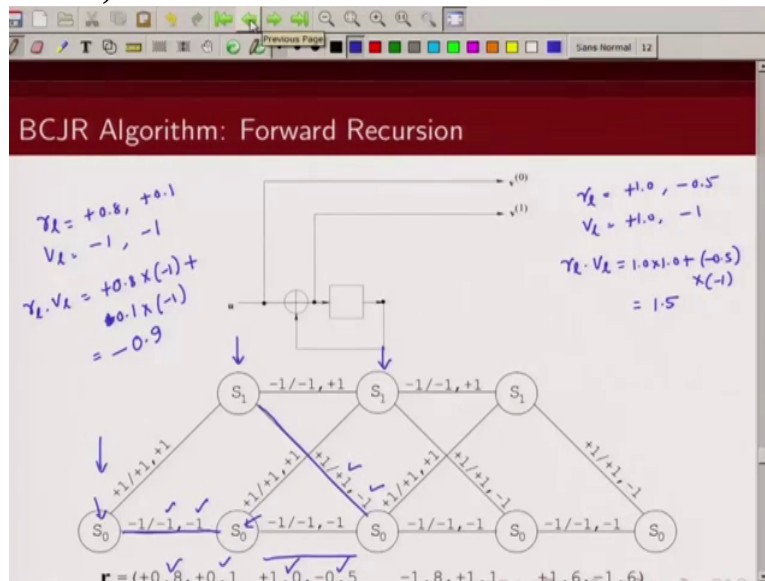
so this is alpha at 0, alpha at 1, so this is alpha at 2. So we are interested to calculate alpha 2 at

(Refer Slide Time 53:39)



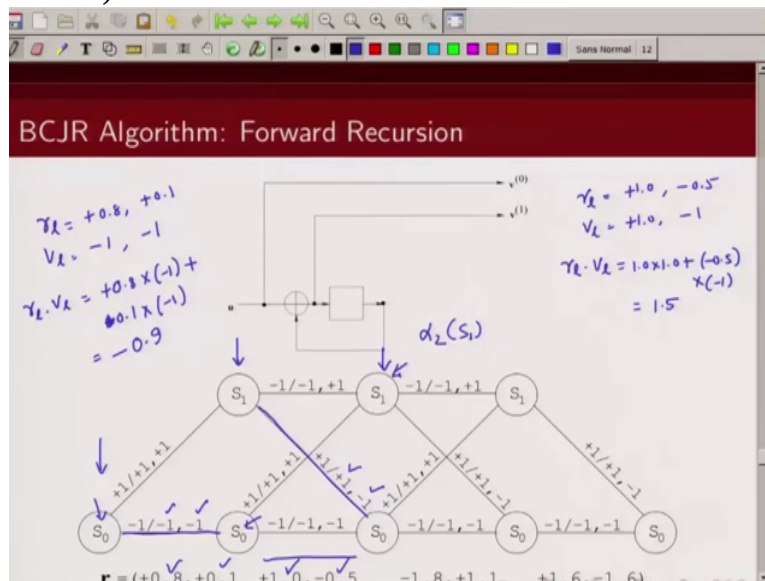
state S 1. So we are interested to calculate

(Refer Slide Time 53:42)



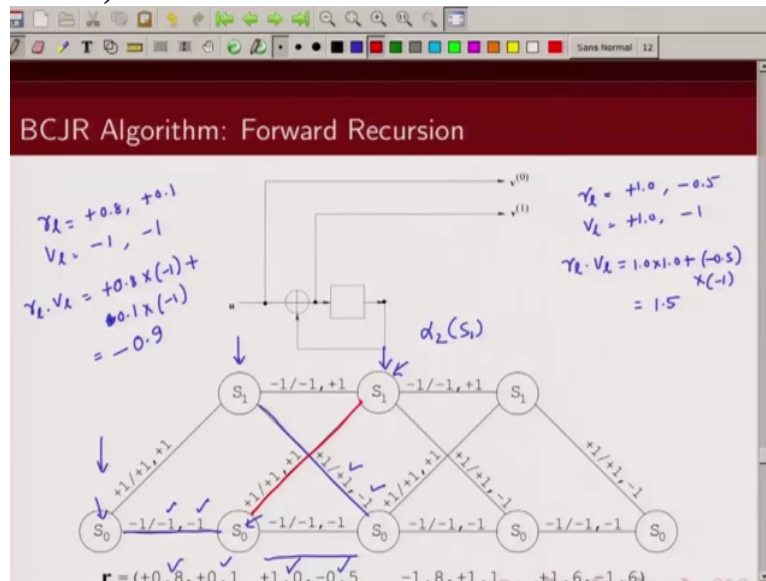
alpha 2 at state S 1. So we are interested to calculate alpha value here.

(Refer Slide Time 53:48)



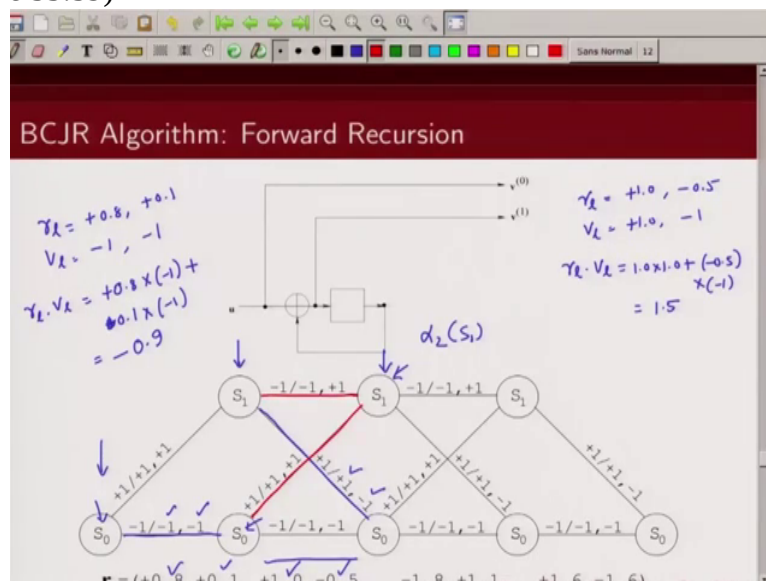
Now what are the transitions that are ending at this state? One is this,

(Refer Slide Time 53:57)



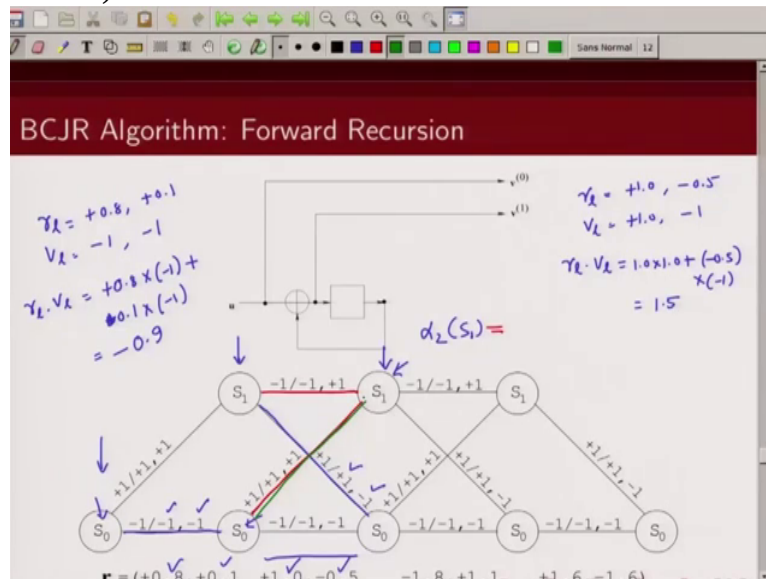
another is this.

(Refer Slide Time 53:59)



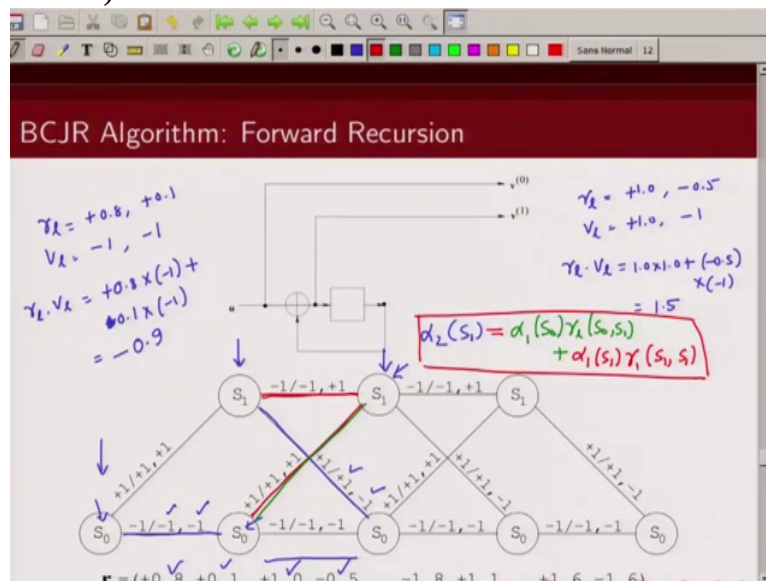
So there will be 2 terms in the alpha computation of this, one corresponding to this transition which is

(Refer Slide Time 54:11)



given by alpha at time 1, S_0 into $\gamma_1 S_0 S_1$ plus there will be another term corresponding to this transition, this will be alpha 1 at state S_1 times gamma 1, initial state S_1 next state S_1 . So this is the value of alpha 2 at state S_1

(Refer Slide Time 54:49)



and that's what we have got here. You can check alpha 1

(Refer Slide Time 54:53)

BCJR Algorithm: Forward Recursion

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

$$\alpha_1(S_0) = \alpha_0(S_0)\gamma_0(S_0, S_0) = 0.6376 \quad (0.2890)$$

$$\alpha_1(S_1) = \alpha_0(S_0)\gamma_0(S_0, S_1) = 1.5683 \quad (0.7110)$$

$$\alpha_2(S_0) = \alpha_1(S_0)\gamma_1(S_0, S_0) + \alpha_1(S_1)\gamma_1(S_1, S_0) = 3.8167 \quad (0.7099)$$

$$\alpha_2(S_1) = \alpha_1(S_0)\gamma_1(S_0, S_1) + \alpha_1(S_1)\gamma_1(S_1, S_1) = 1.5595 \quad (0.2901)$$

$$\alpha_3(S_0) = \alpha_2(S_0)\gamma_2(S_0, S_0) + \alpha_2(S_1)\gamma_2(S_1, S_0) = 5.7821 \quad (0.3824)$$

$$\alpha_3(S_1) = \alpha_2(S_0)\gamma_2(S_0, S_1) + \alpha_2(S_1)\gamma_2(S_1, S_1) = 9.3379 \quad (0.6176)$$

S 0, gamma S 0 S 1, gamma S 0 S 1

(Refer Slide Time 54:57)

BCJR Algorithm: Forward Recursion

Handwritten notes:

$$\gamma_k = +0.8, +0.1$$

$$v_k = -1, -1$$

$$\gamma_k \cdot v_k = +0.8 \times (-1) + 0.1 \times (-1) = -0.9$$

$$\gamma_k = +1.0, -0.5$$

$$v_k = +1.0, -1$$

$$\gamma_k \cdot v_k = 1.0 \times 1.0 + (-0.5) \times (-1) = 1.5$$

$$\alpha_2(S_1) = \alpha_1(S_0)\gamma_1(S_0, S_1) + \alpha_1(S_1)\gamma_1(S_1, S_1)$$

Branch metrics at the bottom: $r = (+0.8, +0.1, +1.0, -0.5, -1.8, +1.1, +1.6, -1.6)$

and the next term is alpha S 1 gamma S 1 S 1,

(Refer Slide Time 55:02)

BCJR Algorithm: Forward Recursion

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

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

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 computed this way. Now we can also normalize the values of alphas. Because alphas are, sum of alphas who are all

(Refer Slide Time 55:22)



state should add up to 1. So in the bracket that you see here,

(Refer Slide Time 55:26)

BCJR Algorithm: Forward Recursion

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

$$\alpha_1(S_0) = \alpha_0(S_0)\gamma_0(S_0, S_0) = \underline{0.6376} \quad (0.2890)$$

$$\alpha_1(S_1) = \alpha_0(S_0)\gamma_0(S_0, S_1) = \underline{1.5683} \quad (0.7110)$$

$$\alpha_2(S_0) = \alpha_1(S_0)\gamma_1(S_0, S_0) + \alpha_1(S_1)\gamma_1(S_1, S_0) = \underline{3.8167} \quad (0.7099)$$

$$\alpha_2(S_1) = \alpha_1(S_0)\gamma_1(S_0, S_1) + \alpha_1(S_1)\gamma_1(S_1, S_1) = \underline{1.5595} \quad (0.2901)$$

$$\alpha_3(S_0) = \alpha_2(S_0)\gamma_2(S_0, S_0) + \alpha_2(S_1)\gamma_2(S_1, S_0) = \underline{5.7821} \quad (0.3824)$$

$$\alpha_3(S_1) = \alpha_2(S_0)\gamma_2(S_0, S_1) + \alpha_2(S_1)\gamma_2(S_1, S_1) = \underline{9.3379} \quad (0.6176)$$

these are the normalized values of alpha. So how do I get this? So this is point 6 3 7 6 by point 6 3 7 6 plus 1 point 5 6 8 3. So this is this

(Refer Slide Time 55:45)

BCJR Algorithm: Forward Recursion

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

$$\alpha_1(S_0) = \alpha_0(S_0)\gamma_0(S_0, S_0) = \underline{0.6376} \quad (0.2890) \quad \frac{0.6376}{0.6376 + 1.5683}$$

$$\alpha_1(S_1) = \alpha_0(S_0)\gamma_0(S_0, S_1) = \underline{1.5683} \quad (0.7110)$$

$$\alpha_2(S_0) = \alpha_1(S_0)\gamma_1(S_0, S_0) + \alpha_1(S_1)\gamma_1(S_1, S_0) = \underline{3.8167} \quad (0.7099)$$

$$\alpha_2(S_1) = \alpha_1(S_0)\gamma_1(S_0, S_1) + \alpha_1(S_1)\gamma_1(S_1, S_1) = \underline{1.5595} \quad (0.2901)$$

$$\alpha_3(S_0) = \alpha_2(S_0)\gamma_2(S_0, S_0) + \alpha_2(S_1)\gamma_2(S_1, S_0) = \underline{5.7821} \quad (0.3824)$$

$$\alpha_3(S_1) = \alpha_2(S_0)\gamma_2(S_0, S_1) + \alpha_2(S_1)\gamma_2(S_1, S_1) = \underline{9.3379} \quad (0.6176)$$

quantity. Similarly this is 1 point 5 6 8 3 divided by this one. So

(Refer Slide Time 55:57)

BCJR Algorithm: Forward Recursion

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

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

Handwritten annotations: 0.6376 (pointing to $\alpha_1(S_0)$), 1.5683 (pointing to $\alpha_1(S_1)$), and $0.6376 + 1.5683 = 2.2063$ (with arrows pointing to the underlined values).

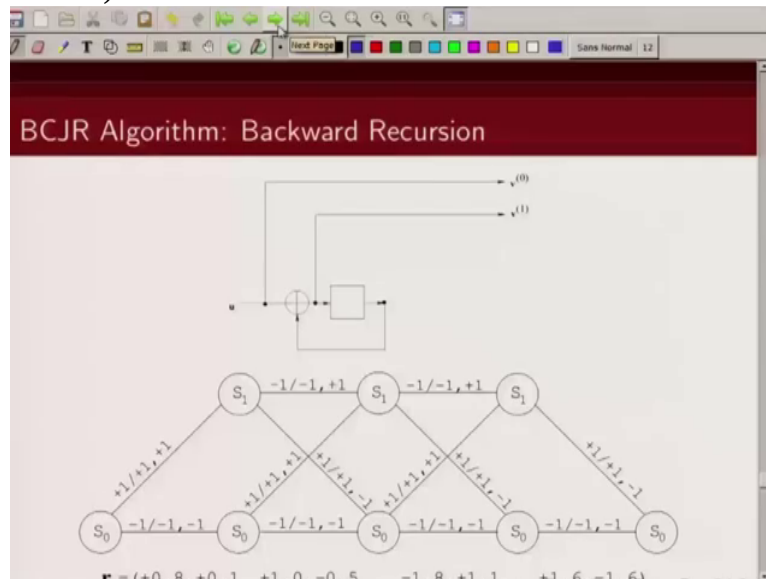
these are actual values of alpha and these are normalized values because these are, sum of probabilities should add up to zero so I can, I can take this value. So when I compute alphas or I can just work with these values or I can work with these values. It is just the scaled version so does not make a difference except for implementation purpose you may want to scale

(Refer Slide Time 56:22)



them, add them up to 1, so that the values do not blow up. Once we have computed

(Refer Slide Time 56:27)



alphas, we can follow the same procedure to compute beta. So

(Refer Slide Time 56:33)

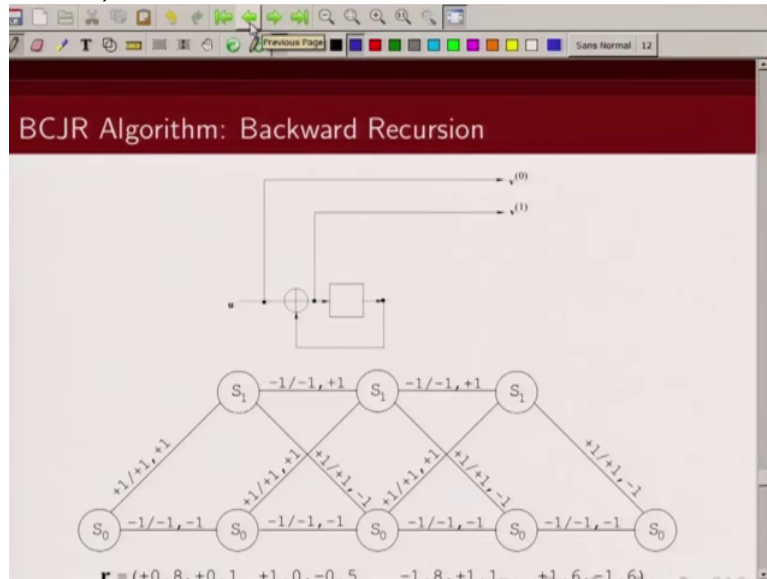
BCJR Algorithm: Backward Recursion

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

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

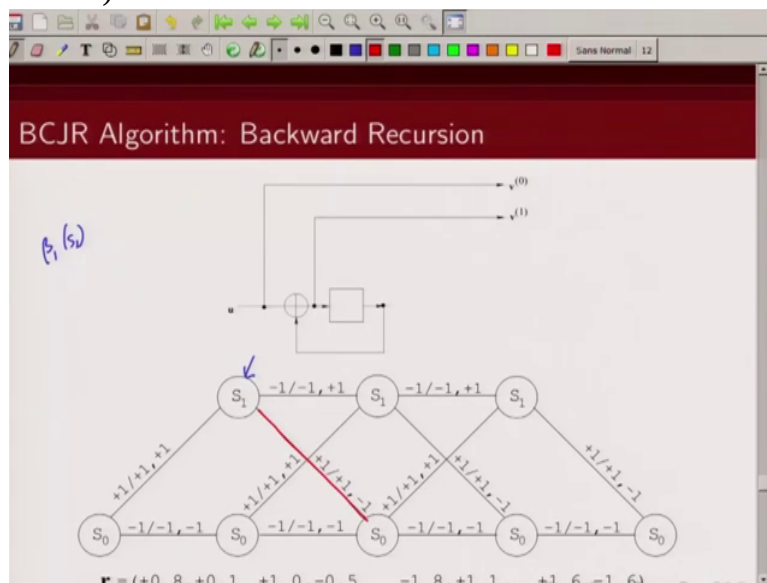
beta computation is given here. Again we can, as an example we can take one particular case. Let's just consider this. beta 1 at state S 1 so this is

(Refer Slide Time 56:45)



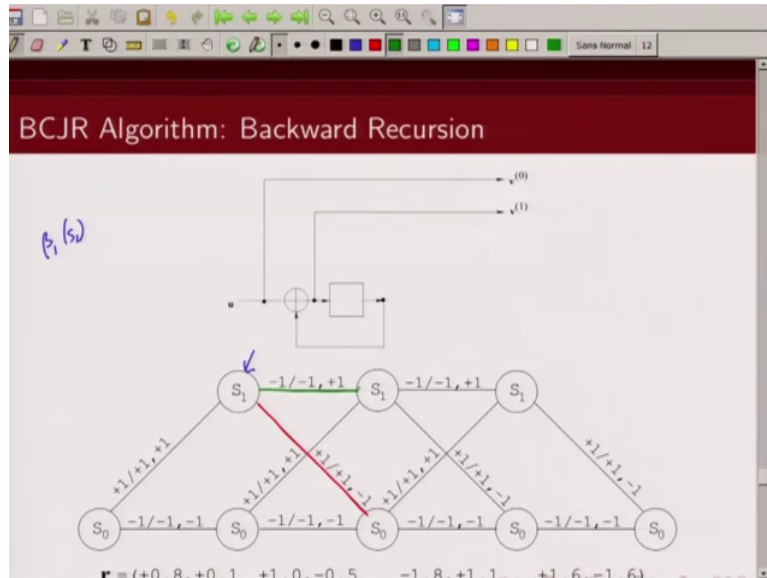
beta 0, beta 1 at state S_1 so we are interested in computing beta 1 at state S_1 . Now what are the transitions that are ending in state S_1 ? One of them is this,

(Refer Slide Time 57:01)



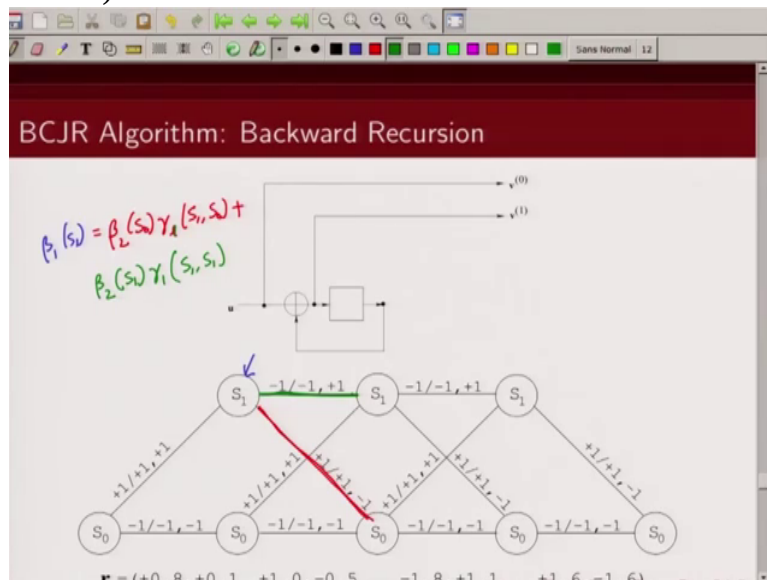
the another one is this. So

(Refer Slide Time 57:05)



what is the contribution from this, this, this transition? This can be written as beta 2 S 0 times gamma 1 S 1 S 0 plus, and the contribution from this will be beta 2 S 1 times gamma of this, this is gamma 1, this is 1, gamma 1 of S 1, S 1. So this, the one in green is corresponding to this transition,

(Refer Slide Time 57:48)



the one in red is due to this transition, Ok, this transition. So beta S 1 can be written as beta 2 S 0 gamma 1 S 1 S 0 plus beta 2 S 1 gamma 1 S 1 S 1. And that's what we have,

(Refer Slide Time 58:06)

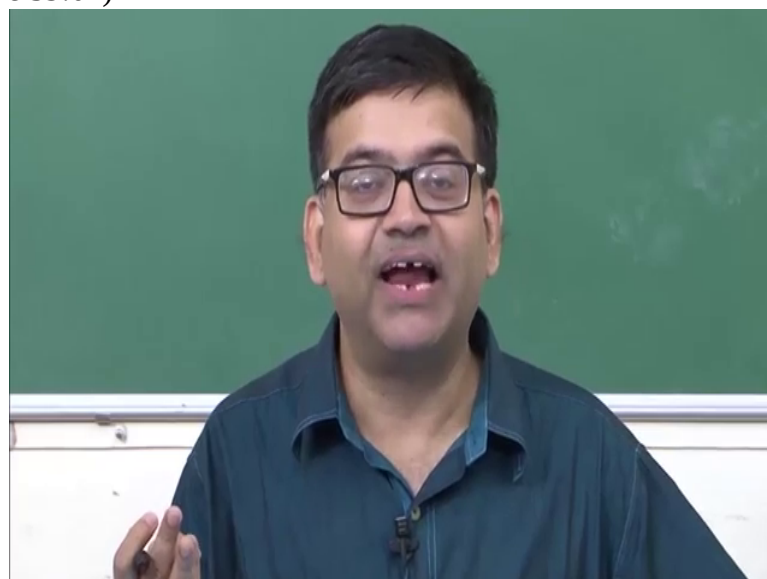
BCJR Algorithm: Backward Recursion

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

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

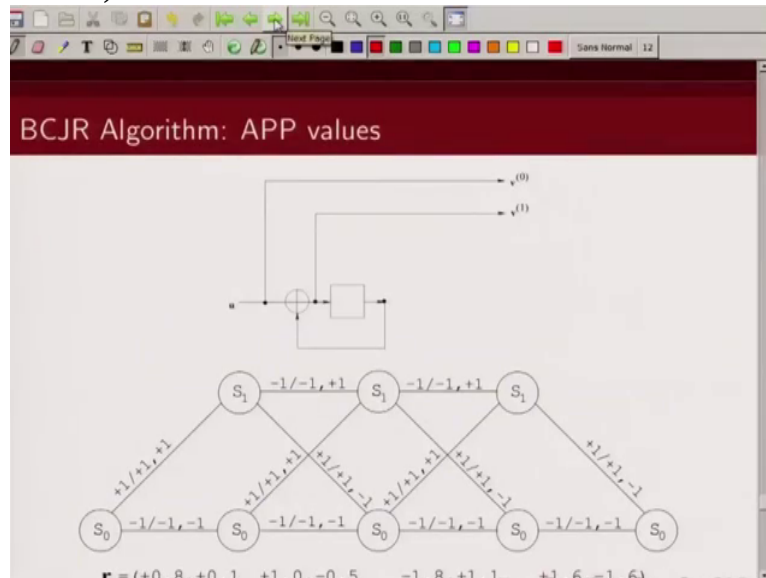
beta 2 S 0 gamma 1 S 1 S 0, beta 2 S 1 gamma 1 S 1 S 1. We already know the values of gammas that we have computed earlier. And we know the values of beta at the end of block. What is this value? This is 1 because the encoder is terminated. And for all other state basically beta 4 S 1 is 0, so this is 1. We know these values so we can compute what is beta 3. Once we know the values of beta 3, we can compute the values of beta 2 because beta 3's values we will require here. Once we know the values of 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 betas and we have the branch metric

(Refer Slide Time 59:04)



gammas. Next step we need to compute the A P P value. And what is the A P P value?

(Refer Slide Time 59:12)



If you recall

(Refer Slide Time 59:14)

The slide titled "BCJR Algorithm: APP values" shows the following steps:

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

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

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

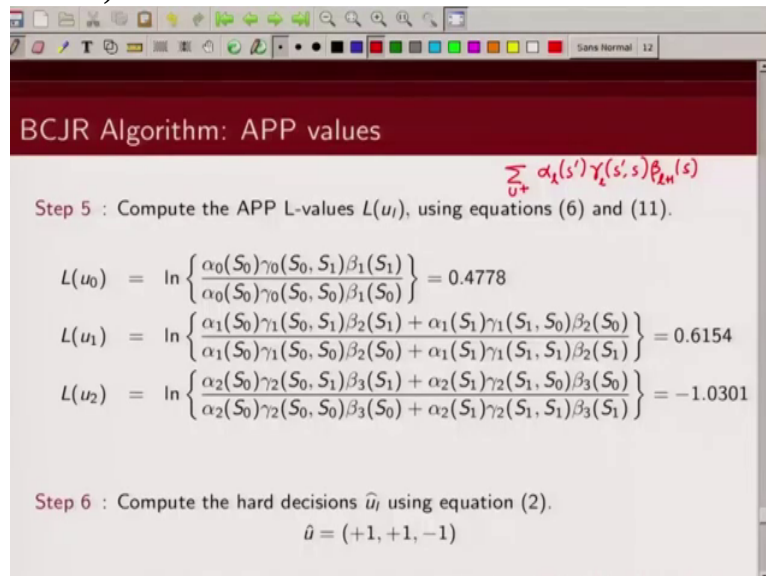
$$L(u_2) = \ln \left\{ \frac{\alpha_2(S_0)\gamma_2(S_0, S_1)\beta_3(S_1) + \alpha_2(S_1)\gamma_2(S_1, S_0)\beta_3(S_0)}{\alpha_2(S_0)\gamma_2(S_0, S_0)\beta_3(S_0) + \alpha_2(S_1)\gamma_2(S_1, S_1)\beta_3(S_1)} \right\} = -1.0301$$

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

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

A P P value, it was product of three terms, alpha, gamma and betas. It's a product of this term. And what was the term in the numerator? We were summing over all those transitions which belongs to u being plus 1 and in the denominator

(Refer Slide Time 59:47)



BCJR Algorithm: APP values

Step 5 : Compute the APP L-values $L(u_i)$, using equations (6) and (11). $\sum_{u'+} \alpha_i(s') \gamma_i(s', s) \beta_{t_n}(s)$

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

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

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

Step 6 : Compute the hard decisions \hat{u}_i using equation (2).
 $\hat{u} = (+1, +1, -1)$

we were summing over all those transitions belonging to u being minus 1. So let's look at how we can compute this. So let's look at first case.

(Refer Slide Time 01:00:12)



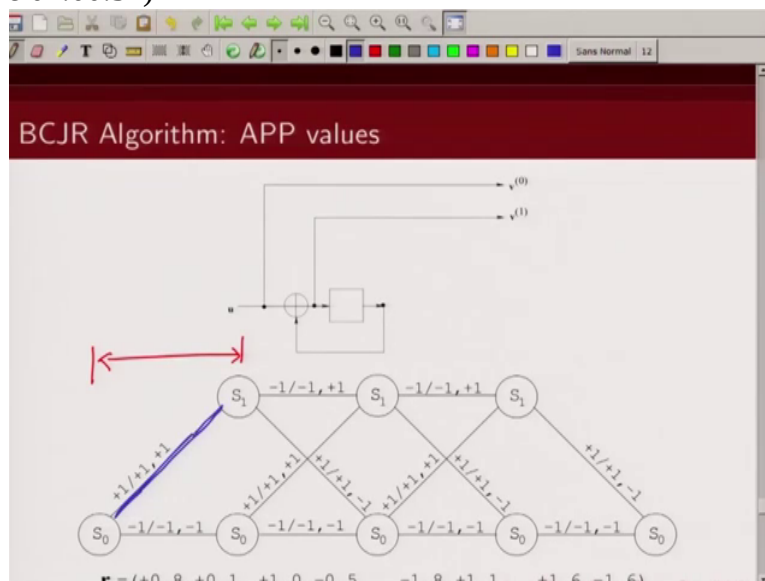
At time, so u_0 is the information sequence that we are trying to estimate for time, first time instance t equal to zero so this is we are looking at this Trellis section. Now which are the transitions corresponding to u_1 being

(Refer Slide Time 01:00:23)



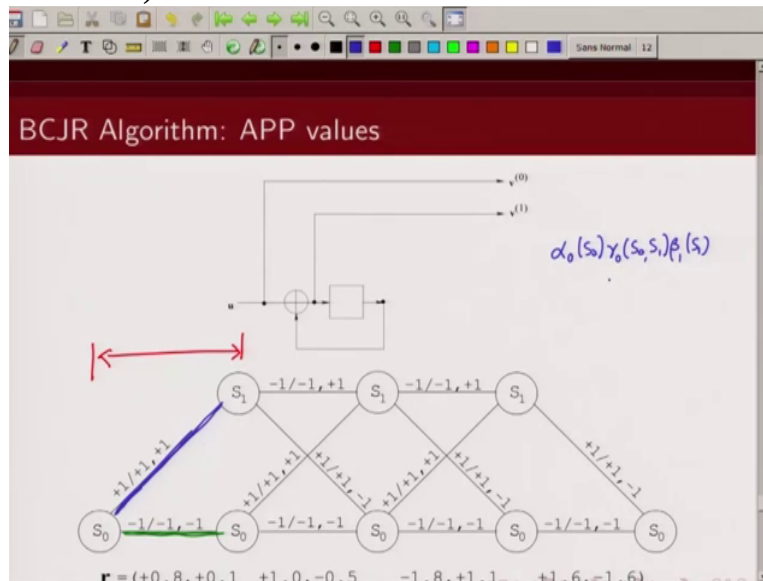
plus 1? So I am denoting by blue, the transitions which is corresponding to

(Refer Slide Time 01:00:31)



u_l being plus 1 and I am denoting by green the transition corresponding to u_l being minus 1, Ok. Then in the numerator then, I will have one term corresponding to this transition and what would be that term? It would be $\alpha_0 S_0$ and γ corresponding

(Refer Slide Time 01:01:14)



to this transition which is $\gamma_0(S_0 \rightarrow S_1)$ times $\beta_1(S_1)$. So I have $\alpha_0(S_0)$, α at this state, γ corresponding to this transition, which is $\gamma_0(S_0 \rightarrow S_1)$, and β corresponding to this state, if we go back what I have,

(Refer Slide Time 01:01:26)

BCJR Algorithm: APP values

$$\sum_{u^*} \alpha_x(s^*) \gamma_x(s^*, s) \beta_{2^n}(s)$$

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

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

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

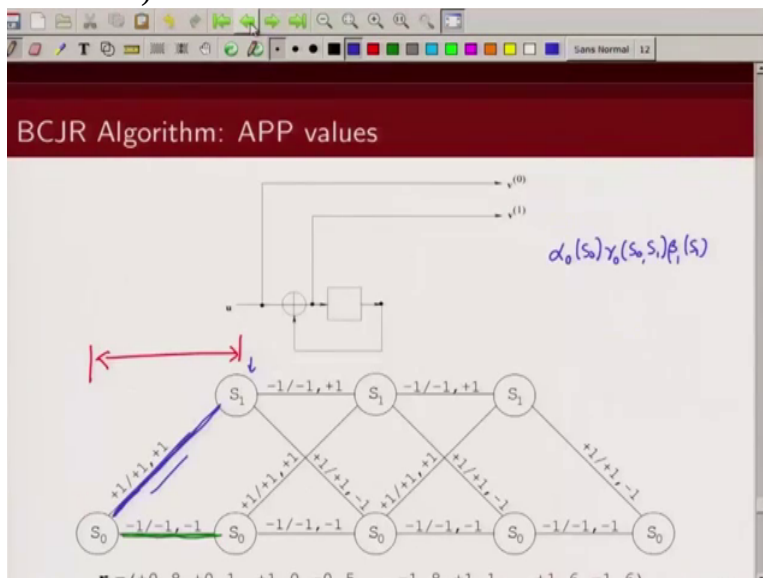
$$L(u_2) = \ln \left\{ \frac{\alpha_2(S_0) \gamma_2(S_0, S_1) \beta_3(S_1) + \alpha_2(S_1) \gamma_2(S_1, S_0) \beta_3(S_0)}{\alpha_2(S_0) \gamma_2(S_0, S_0) \beta_3(S_0) + \alpha_2(S_1) \gamma_2(S_1, S_1) \beta_3(S_1)} \right\} = -1.0301$$

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

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

$\alpha_0(S_0)$, $\gamma_0(S_0 \rightarrow S_1)$ and $\beta_1(S_1)$. At this time instance

(Refer Slide Time 01:01:35)



is there any other transition corresponding to u l plus 1? No. It is only one transition corresponding to u l plus 1. So we will now look at

(Refer Slide Time 01:01:47)

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

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

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

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

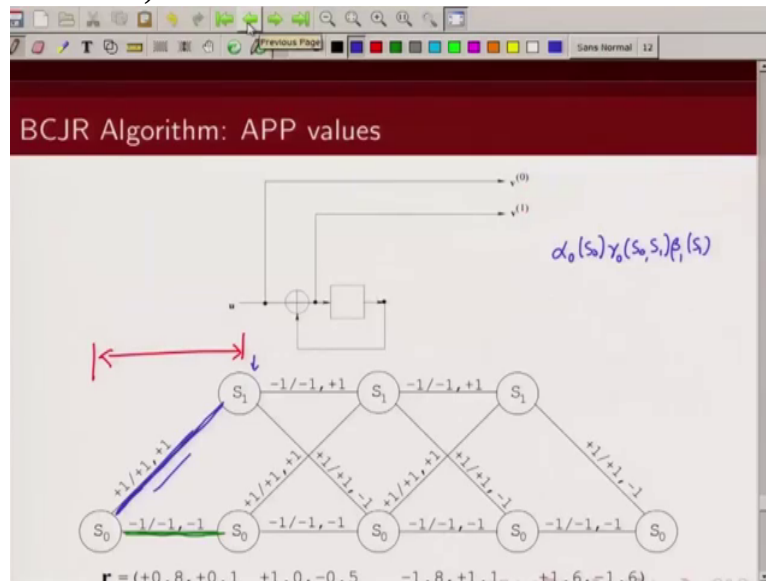
$$L(u_2) = \ln \left\{ \frac{\alpha_2(S_0) \gamma_2(S_0, S_1) \beta_3(S_1) + \alpha_2(S_1) \gamma_2(S_1, S_0) \beta_3(S_0)}{\alpha_2(S_0) \gamma_2(S_0, S_0) \beta_3(S_0) + \alpha_2(S_1) \gamma_2(S_1, S_1) \beta_3(S_1)} \right\} = -1.0301$$

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

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

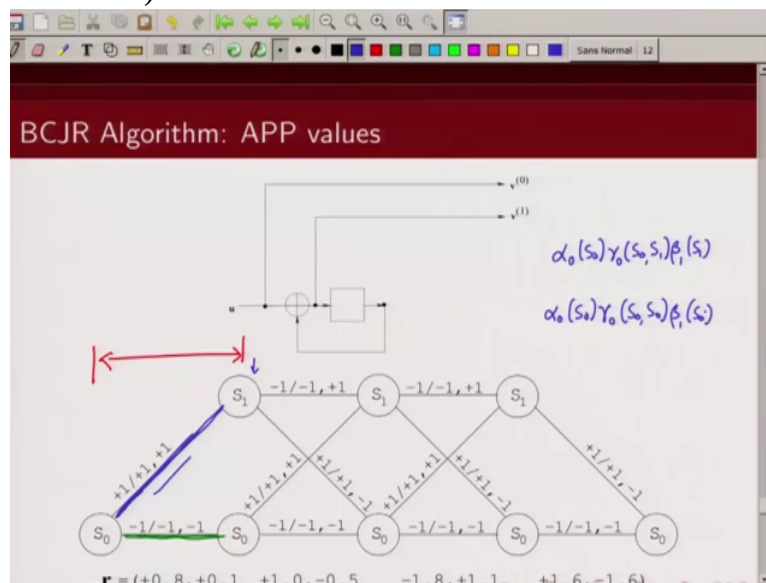
the denominator term. So we have to look for all those transitions corresponding to u being minus 1 and there is only one such

(Refer Slide Time 01:01:56)



transition so the denominator term would be $\alpha_0(s_0)\gamma_0(s_0, s_0)$ and $\beta_1(s_0)$. So this

(Refer Slide Time 01:02:12)



is what we have,

(Refer Slide Time 01:02:13)

BCJR Algorithm: APP values

Step 5 : Compute the APP L-values $L(u_i)$, using equations (6) and (11). $\sum_{u^+} \alpha_i(s') \gamma_i(s', s) \beta_{i,n}(s)$

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

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

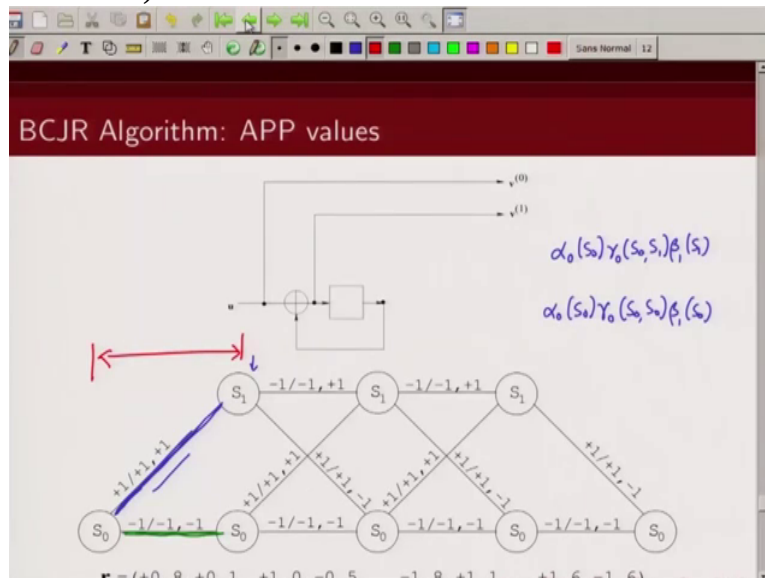
$$L(u_2) = \ln \left\{ \frac{\alpha_2(S_0) \gamma_2(S_0, S_1) \beta_3(S_1) + \alpha_2(S_1) \gamma_2(S_1, S_0) \beta_3(S_0)}{\alpha_2(S_0) \gamma_2(S_0, S_0) \beta_3(S_0) + \alpha_2(S_1) \gamma_2(S_1, S_1) \beta_3(S_1)} \right\} = -1.0301$$

Step 6 : Compute the hard decisions \hat{u}_i using equation (2).
 $\hat{u} = (+1, +1, -1)$

alpha 0 S 0 gamma 0 S 0 S 0 and beta 1 S 0. So we can then calculate the what's the a posteriori L value.

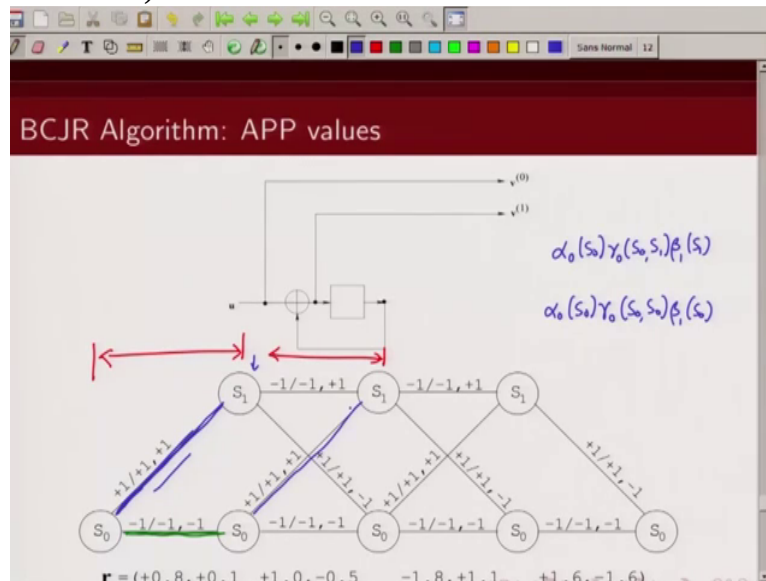
Now let's take another example. Let's take for the second time instance. So the second time instance we are interested in estimating what was our information sequence, information bit. So

(Refer Slide Time 01:02:45)



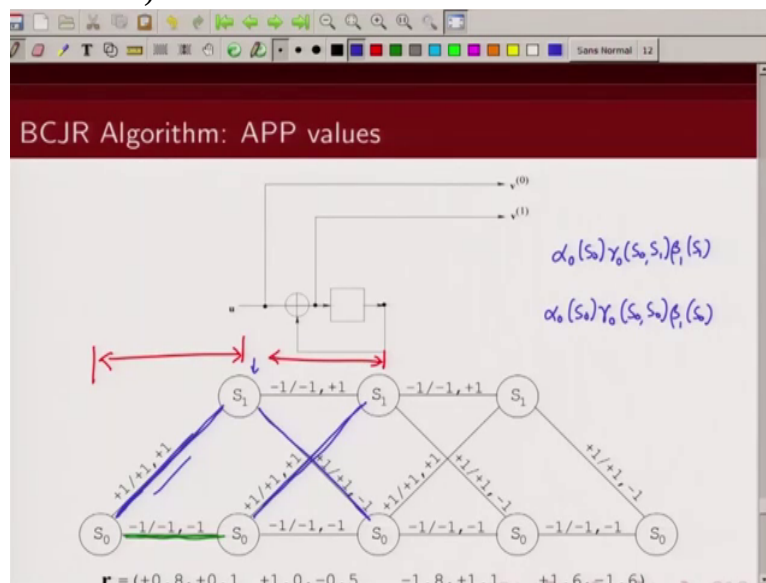
we are now looking at this time instance, this time instance, Ok. Now what are the transitions corresponding to u_1 , information sequence being plus 1? One of them is this. You can see

(Refer Slide Time 01:03:04)



the information sequence is plus 1, that's when you go from S 0 to S 1. And another transition is this one. So

(Refer Slide Time 01:03:15)



these are the two transitions corresponding to u l being plus 1. So in the numerator you will have 2 terms. One corresponding to alpha 1 S 0 gamma 0 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 and that's what you see here.

(Refer Slide Time 01:03:52)

BCJR Algorithm: APP values

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

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

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

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

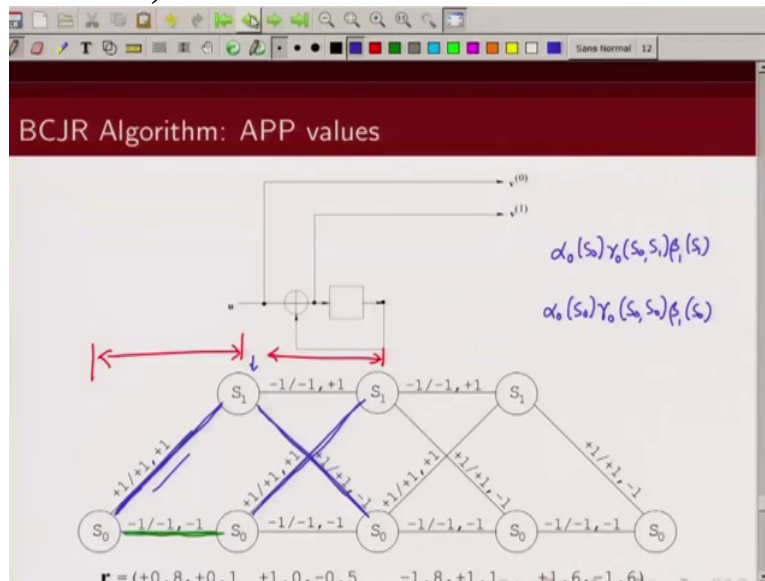
$$L(u_2) = \ln \left\{ \frac{\alpha_2(S_0) \gamma_2(S_0, S_1) \beta_3(S_1) + \alpha_2(S_1) \gamma_2(S_1, S_0) \beta_3(S_0)}{\alpha_2(S_0) \gamma_2(S_0, S_0) \beta_3(S_0) + \alpha_2(S_1) \gamma_2(S_1, S_1) \beta_3(S_1)} \right\} = -1.0301$$

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

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

There are 2 terms, one is alpha 1 S 0 gamma 1 S 0 S 1 beta 2 S 1 this corresponds to this transition and the next term that you see here, There are 2 terms, one is alpha 1 S 0 gamma 1 S 0 S 1 beta 2 S 1 this corresponds to

(Refer Slide Time 01:04:01)



this transition and the next term that you see

(Refer Slide Time 01:04:07)

BCJR Algorithm: APP values

Step 5 : Compute the APP L-values $L(u_i)$, using equations (6) and (11). $\sum_{u^+} \alpha_i(s') \gamma_i(s', s) \beta_{i+1}(s)$

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

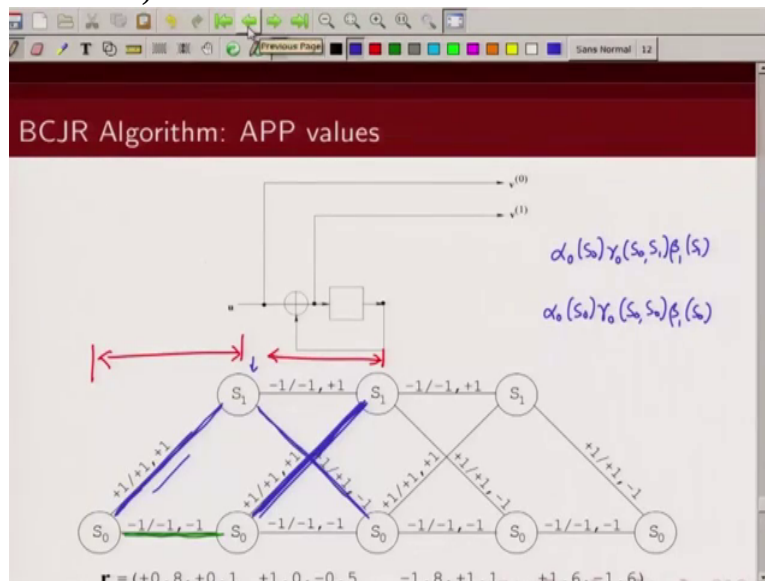
$$L(u_1) = \ln \left\{ \frac{\alpha_1(S_0) \gamma_1(S_0, S_1) \beta_2(S_1) + \alpha_1(S_1) \gamma_1(S_1, S_0) \beta_2(S_0)}{\alpha_1(S_0) \gamma_1(S_0, S_0) \beta_2(S_0) + \alpha_1(S_1) \gamma_1(S_1, S_1) \beta_2(S_1)} \right\} = 0.6154$$

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

Step 6 : Compute the hard decisions \hat{u}_i using equation (2).
 $\hat{u} = (+1, +1, -1)$

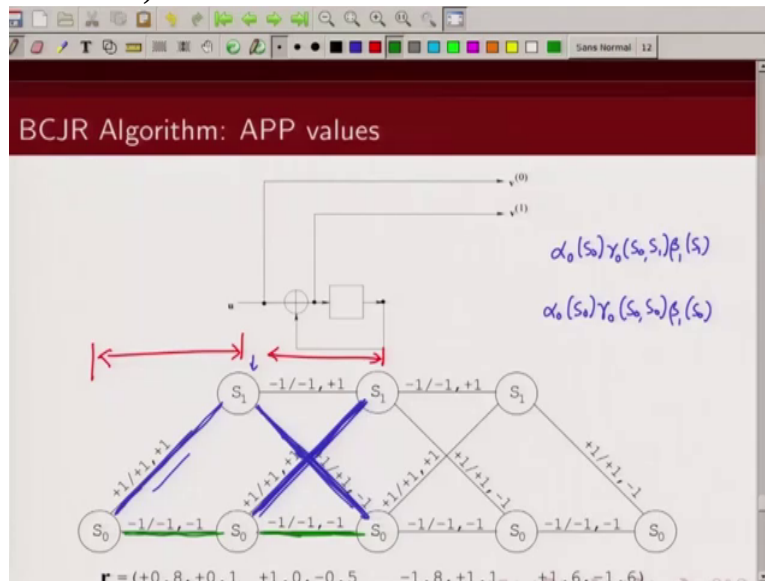
here, this one corresponds to

(Refer Slide Time 01:04:11)



this transition, Ok. Now similarly in the denominator you need to look at what are the valid transitions corresponding to u_1 minus 1 and what are those? One of them is this

(Refer Slide Time 01:04:29)



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 plus alpha 1 S 1 gamma 1 S 1 S 1 times beta 2 S 1. And that's what you have here.

(Refer Slide Time 01:05:04)

BCJR Algorithm: APP values

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).

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

So likewise we compute log likelihood ratios, APP values for all the three information bits. Now what is the final step? Once we have computed the log likelihood ratio we will see whether these log likelihood ratios are greater than 0 or less than 0. If they are greater than equal to 0, we decide in favor of u_i being plus 1, otherwise we decide in favor of u_i being minus 1. So this is point 4 7 7 8, which is greater than zero so we decide in favor of plus 1.

This is greater than zero so we decide in favor of plus 1 and this one is less than zero so we decide in favor of minus 1. So then the final decoded bits are plus 1,

(Refer Slide Time 01:06:00)

BCJR Algorithm: APP values

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).

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

plus 1 and minus 1. So with this I will conclude this lecture, thank you.

(Refer Slide Time 01:06:08)

