An introduction to coding theory

Adrish Banerjee

Department of Electrical Engineering
Indian Institute of Technology Kanpur
Kanpur, Uttar Pradesh
India

Feb. 20, 2017

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

BCJR Algorithm

Outline of the lecture

- BCJR algorithm for convolutional codes

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.

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.

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.

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.

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

BCJR Algorithm

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

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

- The decoder output is given by

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

BCJR Algorithm

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

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

where

BCJR Algorithm

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

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

where

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

BCJR Algorithm

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

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

where

- $\mathbf{U}_l^+$ is the set of all information sequences $\mathbf{u}$ such that $u_l = +1$,
- $\mathbf{v}$ is the transmitted codeword corresponding to the information sequence $\mathbf{u}$, and

BCJR Algorithm

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

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

where

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

Adrish Banerjee Department of Electrical Engineering Indian Institute of Technology Kanpur Kanpur, Uttar Pradesh India
An introduction to coding theory

BCJR Algorithm

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

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

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

BCJR Algorithm

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

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

  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.

BCJR Algorithm

- Equation (3) can be re-written as

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

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.

BCJR Algorithm

- Equation (3) can be re-written as

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

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

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

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.

BCJR Algorithm

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

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

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

BCJR Algorithm

- Defining

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

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

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

BCJR Algorithm

- Defining

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

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

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

- Equation (7) can be written as

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

BCJR Algorithm

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

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

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

BCJR Algorithm

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

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

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

BCJR Algorithm

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

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

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

BCJR Algorithm

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

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

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

BCJR Algorithm

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

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

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

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

Adrish Banerjee Department of Electrical Engineering Indian Institute of Technology Kanpur Kanpur, Uttar Pradesh India
An introduction to coding theory

BCJR Algorithm

Initial conditions for recursion

- Forward recursion:

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

BCJR Algorithm

Initial conditions for recursion

- Forward recursion:

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

- Backward recursion:

$$\beta_K(s) = \begin{cases} 1, & s = \mathbf{0} \\ 0, & s \neq \mathbf{0} \end{cases}, \tag{17}$$

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 $\gamma_l(s', s)$, $l = 0, 1, \cdots, K-1$, using equation (14).

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

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

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

Step 6 : Compute the hard decisions $\widehat{u}_l$ using equation (2).

BCJR Algorithm

Example:

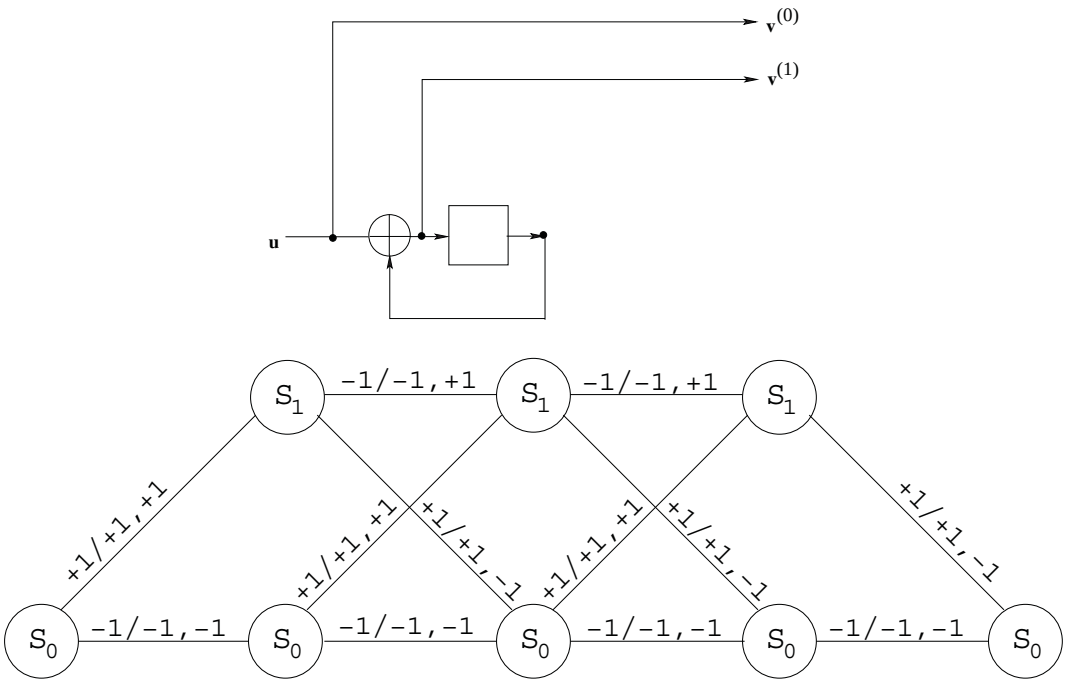
- Consider the $(2, 1, 1)$ systematic recursive convolutional code with generator matrix

$$\mathbf{G}(D) = [1 \quad 1/(1+D)]$$

- 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

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

BCJR Algorithm



BCJR Algorithm

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

Initial conditions for recursion

- Forward recursion:

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

BCJR Algorithm

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

Initial conditions for recursion

- Forward recursion:

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

- Backward recursion:

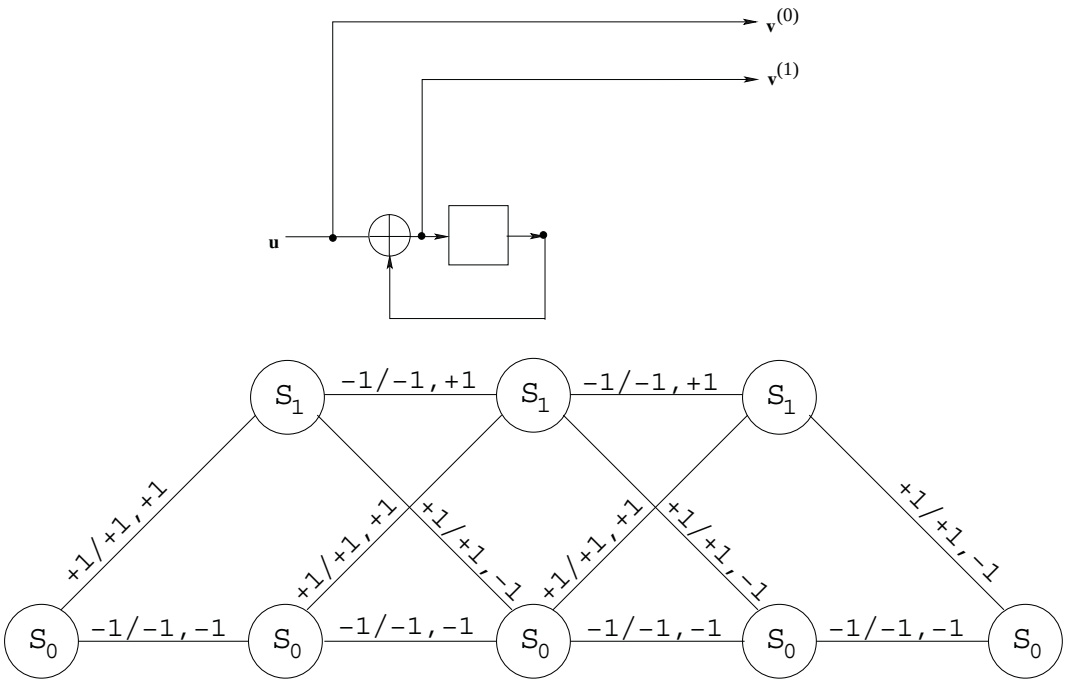
$$\beta_K(s) = \begin{cases} 1, & s = \mathbf{0} \\ 0, & s \neq \mathbf{0} \end{cases},$$

BCJR Algorithm: Example

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

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

BCJR Algorithm: Forward Recursion

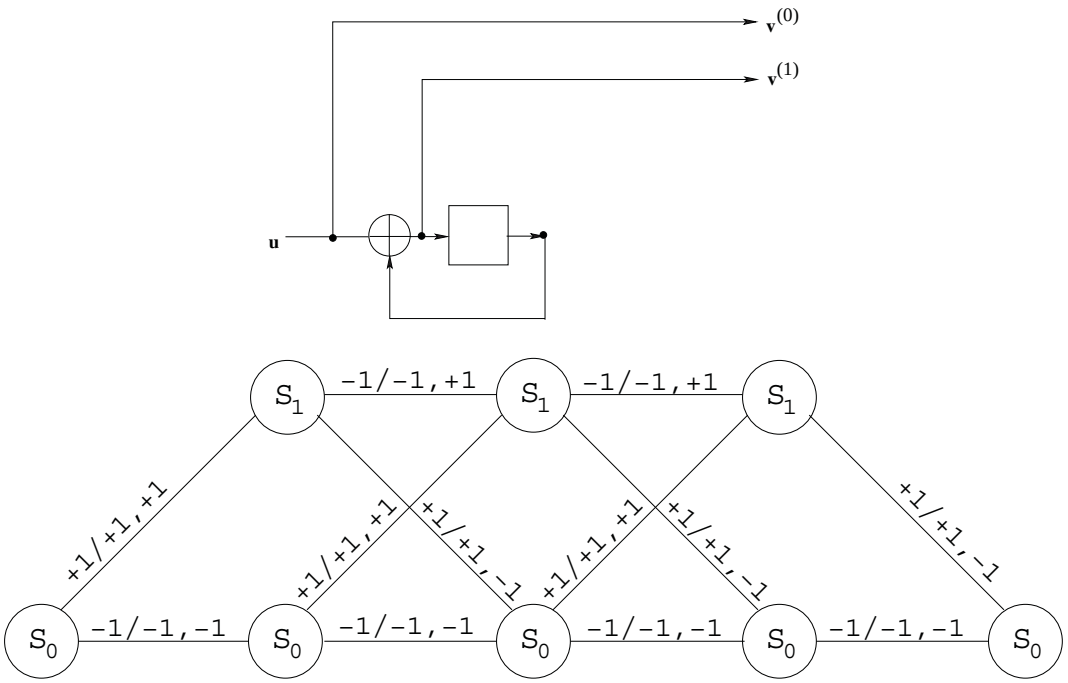


BCJR Algorithm: Forward Recursion

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

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

BCJR Algorithm: Backward Recursion



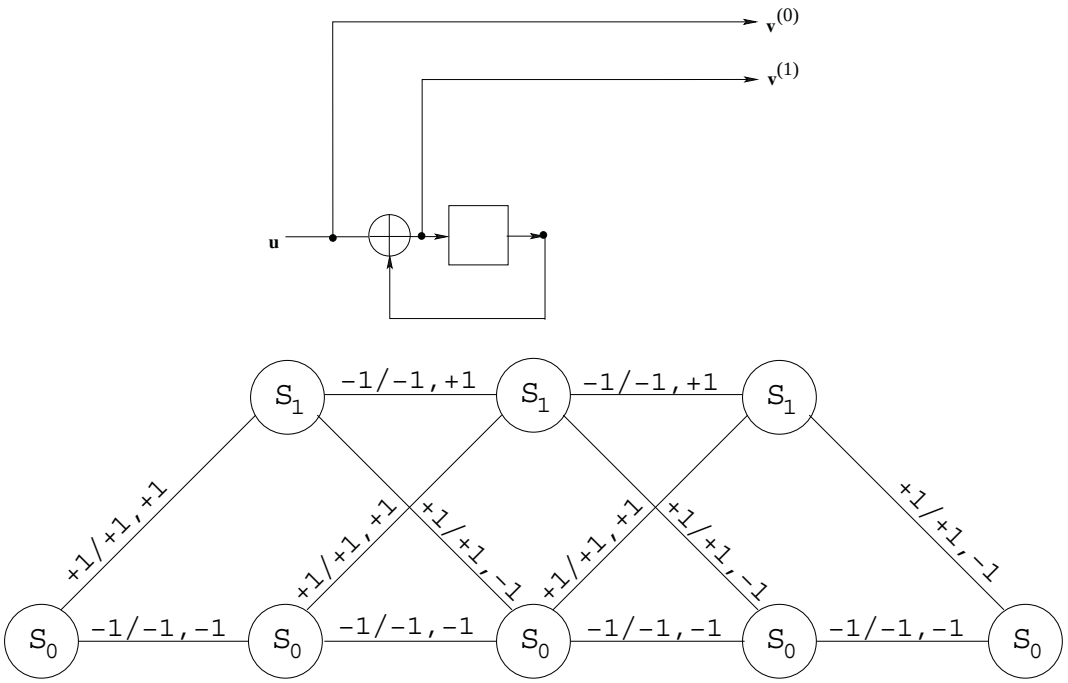
$\mathbf{r} = (+0.8, +0.1 \quad +1.0, -0.5 \quad -1.8, +1.1 \quad +1.6, -1.6)$

BCJR Algorithm: Backward Recursion

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

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

BCJR Algorithm: APP values



$\mathbf{r} = (+0.8, +0.1 \quad +1.0, -0.5 \quad -1.8, +1.1 \quad +1.6, -1.6)$

BCJR Algorithm: APP values

Step 5 : Compute the APP L-values $L(u_l)$, 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 $\widehat{u}_l$ using equation (2).

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