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
Outline of the lecture

- BCJR algorithm for convolutional codes

To minimize the bit error rate (BER), the a-posteriori probability \( P(\hat{u}_i = u_i | r) \) that an information bit \( u_i \) is correctly decoded must be maximized.
BCJR Algorithm

- To minimize the bit error rate (BER), the a-posteriori probability $P(\hat{u}_i = u_i | r)$ that an information bit $u_i$ is correctly decoded must be maximized.
- An algorithm that maximizes $P(\hat{u}_i = u_i | r)$ 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.
To minimize the bit error rate (BER), the a-posteriori probability $P(\hat{u}_l = u_l | r)$ that an information bit $u_l$ is correctly decoded must be maximized.

An algorithm that maximizes $P(\hat{u}_l = u_l | r)$ 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.

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

The decoder output is given by

$$\hat{u}_l = \begin{cases} 
+1 & \text{if } L(u_l) > 0 \\
-1 & \text{if } L(u_l) < 0
\end{cases}, \quad l = 0, 1, \cdots, k - 1. \hspace{1cm} (2)$$
The APP value $P(u_l = +1|r)$ as follows:

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

where

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

Adrish Banerjee
Department of Electrical Engineering
Indian Institute of Technology Kanpur
Kanpur, Uttar Pradesh India
An introduction to coding theory
The APP value $P(u_l = +1|r)$ as follows:

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

where

- $U^+_l$ is the set of all information sequences $u$ such that $u_l = +1$,
- $v$ is the transmitted codeword corresponding to the information sequence $u$, and
- $p(r|v)$ is the pdf of the received sequence $r$ given $v$. 
The expression in equation (1) for the APP L-value becomes

\[
L(u_i) = \ln \left[ \frac{\sum_{u \in U_i^+} p(r|v)P(u)}{\sum_{u \in U_i^-} p(r|v)P(u)} \right],
\]

(4)

where \( U_i^- \) is the set of all information sequences \( u \) 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.
BCJR Algorithm

Equation (3) can be re-written as

\[ P(u_l = +1 | r) = \frac{p(u_l = +1, r)}{P(r)} = \sum_{(s', s) \in \Sigma_i^+} \frac{p(s_l = s', s_{l+1} = s, r)}{P(r)}, \]

where \( \Sigma_i^+ \) 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

\[ L(u_l) = \ln \left\{ \frac{\sum_{(s', s) \in \Sigma_i^+} p(s_l = s', s_{l+1} = s, r)}{\sum_{(s', s) \in \Sigma_i^-} p(s_l = s', s_{l+1} = s, r)} \right\}, \]

where \( \Sigma_i^- \) 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 \( l \).
The joint pdf’s $p(s', s, r)$ in equation (6) can be evaluated recursively

$$p(s', s, r) = p(s', s, r_{t<1}, r_l, r_{t>l})$$

$$= p(r_{t>1}|s', s, r_{t<1}, r_l)p(s', s, r_{t<1}, r_l)$$

$$= p(r_{t>1}|s', s, r_{t<1}, r_l)p(s, r_l|s', r_{t<1})p(s', r_{t<1})$$

$$= p(r_{t>1}|s)p(s, r_l|s')p(s', r_{t<1})$$  

(7)

where $r_{t<1}$ represents the portion of the received sequence $r$ before time $l$ and $r_{t>1}$ represents the portion of the received sequence $r$ after time $l$.

Defining

$$\alpha_l(s') \equiv p(s', r_{t<1})$$

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

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

(8)  

(9)  

(10)
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)

Equation (7) can be written as

\[ p(s', s, r) = \beta_{l+1}(s)\gamma_l(s', s)\alpha_l(s'). \] (11)

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

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

where \( \sigma_l \) is the set of all states at time \( l \).
Similarly expression or the probability $\beta_l(s')$ can be written as

$$\beta_l(s') = p(r_{t>(l-1)}|s')$$

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

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

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

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

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

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

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

$$\gamma_l(s', s) = p(s, r_l|s') = \frac{p(s', s, r_l)}{P(s')}$$

$$= \left[ \frac{P(s', s)}{P(s')} \right] \left[ \frac{p(s', s, r_l)}{P(s', s)} \right]$$

$$= P(s|s')p(r_l|s', s) = P(u_l)p(r_l|v_l),$$

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

Adrish Banerjee  
Department of Electrical Engineering  
Indian Institute of Technology Kanpur  
Kanpur, Uttar Pradesh India
For a continuous output AWGN channel, if $s' \rightarrow s$ is a valid state transition,

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

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.
BCJR Algorithm

Initial conditions for recursion

- Forward recursion:

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

- Backward recursion:

\[
\beta_K(s) = \begin{cases} 
1, & s = 0 \\
0, & s \neq 0 
\end{cases}, \quad (17)
\]
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 $\hat{u}_l$ using equation (2).

Example:

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

$$G(D) = \begin{bmatrix} 1 & 1/(1 + D) \end{bmatrix}$$

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

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

$$= (+0.8, +0.1; +1.0, -0.5; -1.8, +1.1; +1.6, -1.6).$$
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:
  $\alpha_0(s) = \begin{cases} 
  1, & s = 0 \\
  0, & s \neq 0 
  \end{cases}$
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**:
  
  $\beta_K(s) = \begin{cases} 
  1, & s = 0 \\
  0, & s \neq 0
  \end{cases}$

BCJR Algorithm: Example

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

\[
\begin{align*}
\gamma_0(S_0, S_0) &= e^{-0.45} = 0.6376 \\
\gamma_0(S_0, S_1) &= e^{0.45} = 1.5683 \\
\gamma_1(S_0, S_0) &= e^{-0.25} = 0.7788 \\
\gamma_1(S_0, S_1) &= e^{0.25} = 1.2840 \\
\gamma_1(S_1, S_1) &= e^{-0.75} = 0.4724 \\
\gamma_1(S_1, S_0) &= e^{0.75} = 2.1170 \\
\gamma_2(S_0, S_0) &= e^{0.35} = 1.4191 \\
\gamma_2(S_0, S_1) &= e^{-0.35} = 0.7047 \\
\gamma_2(S_1, S_1) &= e^{1.45} = 4.2631 \\
\gamma_2(S_1, S_0) &= e^{-1.45} = 0.2346 \\
\gamma_3(S_0, S_0) &= e^0 = 1.0 \\
\gamma_3(S_1, S_0) &= e^{1.6} = 4.9530
\end{align*}
\]
Step 3: Compute the forward metrics $\alpha_{l+1}(s)$, $l = 0, 1, \cdots, K - 1$, using equation (12).

\begin{align*}
\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{align*}
Step 4 : Compute the backward metrics $\beta_l(s')$, $l = K - 1, K - 2, \ldots, 0$, using equation (13).

\[
\begin{align*}
\beta_3(S_0) &= \beta_4(S_0)\gamma_3(S_0, S_0) = 1.0 \ (0.1680) \\
\beta_3(S_1) &= \beta_4(S_0)\gamma_3(S_1, S_0) = 4.9530 \ (0.8320) \\
\beta_2(S_0) &= \beta_3(S_0)\gamma_2(S_0, S_0) + \beta_3(S_1)\gamma_2(S_0, S_1) = 4.9095 \ (0.1870) \\
\beta_2(S_1) &= \beta_3(S_0)\gamma_2(S_1, S_0) + \beta_3(S_1)\gamma_2(S_1, S_1) = 21.3497 \ (0.8130) \\
\beta_1(S_0) &= \beta_2(S_0)\gamma_1(S_0, S_0) + \beta_2(S_1)\gamma_1(S_0, S_1) = 31.2365 \ (0.6040) \\
\beta_1(S_1) &= \beta_2(S_0)\gamma_1(S_1, S_0) + \beta_2(S_1)\gamma_1(S_1, S_1) = 20.4790 \ (0.3960)
\end{align*}
\]
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 $\hat{u}_l$ using equation (2).

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