The Viterbi Algorithm
EECS 869: Error Control Coding
Fall 2013

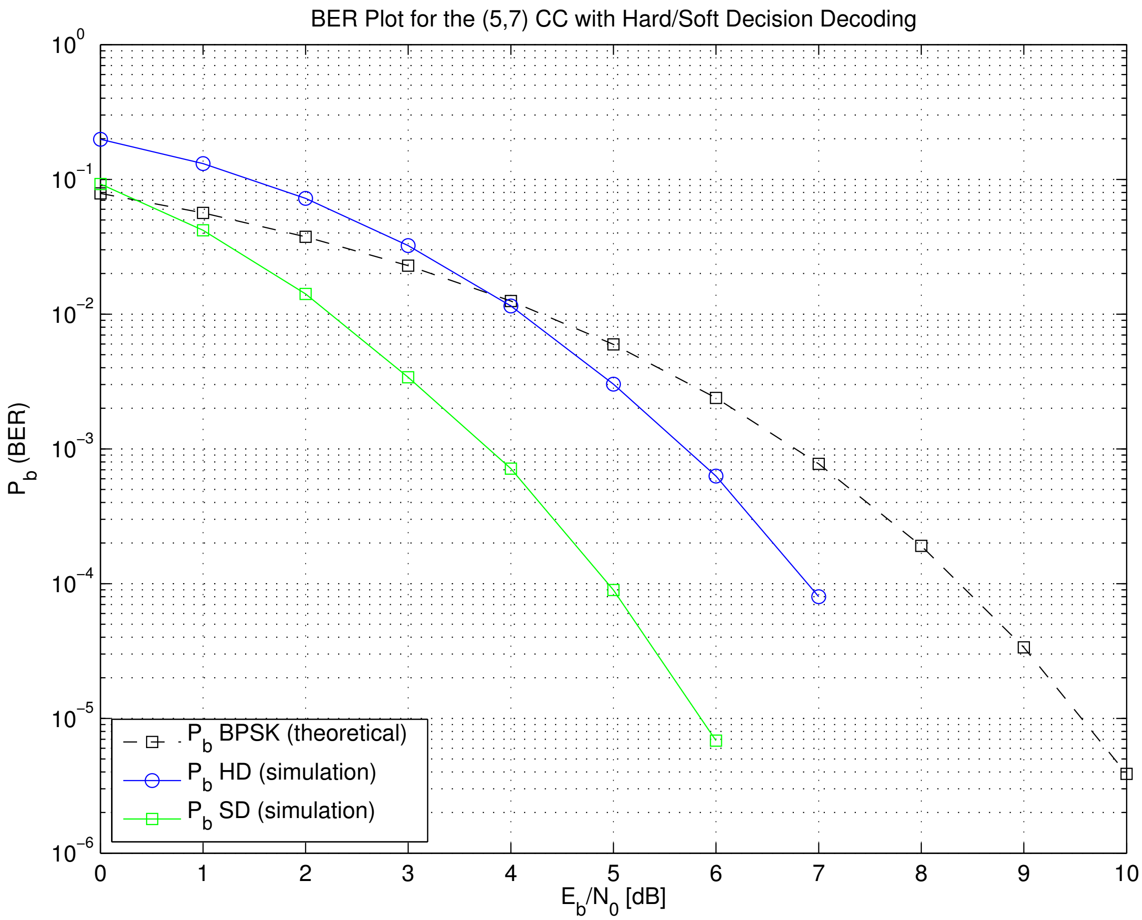
1 Background Material

Complete the following tasks. You should submit an e-mail with three .m file attachments (use the e-mail address esp@eecs.ku.edu).

1. **Implement a Function to Create Look-up Tables for the Trellis for Convolutional Codes.** You may assume that we are limited to rate $1/n$ feedforward non-systematic codes. You need to create look-up tables that are sorted according to the right-hand edge index, e^{R}; tables are needed for the left-hand edge index, e^{L}, the starting state, s^{S}, the message bit, $\mathbf{m}(e)$, the code bits, $\mathbf{c}(e)$, and the ending state, s^{E}. You should implement this as a MATLAB functionwith the following syntax:
   ```
   [eL, sS, me, ce, sE] = CreateCcTrellisXXX(G);
   ```
   where $\mathbf{G}$ is a $(\nu+1)\times n$ matrix that specifies n encoder functions with a constraint length (memory order) of ν. For the (5,7) code, we have a transfer function matrix
   $$\mathbf{G}(D) = \begin{bmatrix} 1+D^2 & 1+D+D^2 \end{bmatrix}$$
   and so we specify $\mathbf{G}$ in MATLAB as
   $$\mathbf{G} = \begin{bmatrix} 1 & 1 \\ 0 & 1 \\ 1 & 1 \end{bmatrix}.$$
   As for the output arguments, each is an $N_{\mathrm{E}}\times 1$ MATLAB vector (you should implement ce as a $N_{\mathrm{E}}\times n$ matrix), where $N_{\mathrm{E}}\times 1$ is the number of edges in the trellis. Because these elements are in order of increasing right-hand edge index, the specifiction for e^{R} is implied and does not need to be stated explicitly.

2. **Implement the Hard-Decision Viterbi Algorithm for Convolutional Codes.** You should implement this as a MATLAB function with the following syntax:
   ```
   m_hat = VaCcHdXXX(r, TERM, sS, me, ce);
   ```
   where $\mathbf{r}$ is a $1\times nL$ MATLAB vector containing hard decisions from the BSC (0's and 1's) and TERM is a Boolean flag (i.e., a 0 or a 1) to indicate if the trellis is terminated. The metric increment for the hard-decision VA is
   $$\gamma_t(e) = d_{\mathrm{H}}\big(\mathbf{r}_t, \mathbf{c}(e)\big)$$
   where $d_{\mathrm{H}}(\cdot,\cdot)$ denotes Hamming distance. For the (5,7) code, the metric increment is 0, 1, or 2. The objective of the VA is to *minimize* this metric. The algorithm you need to implement is shown in Algorithm 2.

3. **Implement the Soft-Decision Viterbi Algorithm for Convolutional Codes.** You should implement this as a MATLAB function with the following syntax:
   ```
   m_hat = VaCcSdXXX(r, TERM, sS, me, ae);
   ```



where $\mathbf{r}$ is a $1 \times nL$ MATLAB vector containing soft decisions from the AWGN channel (noisy +1's and −1's). The metric increment for the soft-decision VA is

$$\gamma_t(e) = \sum_{i=0}^{n-1} r_t^{(i)} a^{(i)}(e) = \mathbf{a}(e)\mathbf{r}_t^T$$

where $\mathbf{a}(e)$ is the antipodal version of $\mathbf{c}(e)$ and $(\cdot)^T$ is the transpose operator. Over time, the increments $\mathbf{r}_t\mathbf{a}^T(e)$ add up and result in the correlation between $\mathbf{r}$ and $\mathbf{a}$. The objective of the VA is to *maximize* the correlation, so the appropriate changes must be made to the VA operations. The algorithm you need to implement is shown in Algorithm 3.

4. **BER Simulation.** Verify the correct operation of your VA implementations by running a BER simulation (using the "wrapper file" provided) for the (5,7) code. The expected result is shown above. You should run your simulation over the range of E_b/N_0 values from 0 to 5 dB.

Algorithm 1 The Viterbi algorithm (VA) for hard-decision inputs.

Input: The received signal $\mathbf{r} = \{\mathbf{r}_0, \mathbf{r}_1, \cdots, \mathbf{r}_{L-1}\}$.
Input: A Boolean flag TERM to indicate if the trellis is terminated.
Input: Trellis look-up tables indexed by the generic edge index e: $s^{\mathrm{S}}(e)$, $\mathbf{m}(e)$, and $\mathbf{c}(e)$.
Output: The estimated symbol sequence $\hat{\mathbf{m}} = \{\hat{\mathbf{m}}_0, \hat{\mathbf{m}}_1, \cdots, \hat{\mathbf{m}}_{L-1}\}$.
Initialization:
if TERM == TRUE **then**,
    $A_{-1}(0) = 0$;
    $A_{-1}(s) = +\infty$ for $1 \le s \le N_{\mathrm{S}} - 1$;
else
    $A_{-1}(s) = 0$ for $0 \le s \le N_{\mathrm{S}} - 1$;
end if;
Main Algorithm:
// Forward recursion with metric and survivor updates
for $t = 0, 1, \cdots, L-1$ **do**
    for $s = 0, 1, \cdots, N_{\mathrm{S}} - 1$ **do**
        $A_t(s) = \min\limits_{e: s^{\mathrm{E}}(e)=s} \left\{ A_{t-1}(s^{\mathrm{S}}(e)) + \gamma_t(e) \right\}$;
        $T_t(s) = \arg\min\limits_{e: s^{\mathrm{E}}(e)=s} \left\{ A_{t-1}(s^{\mathrm{S}}(e)) + \gamma_t(e) \right\}$;
        where $\gamma_t(e) = d_{\mathrm{H}}\big(\mathbf{r}_t, \mathbf{c}(e)\big)$;
    end for
end for
// Final global survivor at the end of the transmission
if TERM == TRUE **then**,
    $\hat{s}^{\mathrm{E}}_{L-1} = 0$;
else
    $\hat{s}^{\mathrm{E}}_{L-1} = \arg\min\limits_{0 \le s \le N_{\mathrm{S}}-1} \{A_{L-1}(s)\}$;
end if;
// Traceback operation to identify output sequence
for $t = L-1, \cdots, 1, 0$ **do**
    $\hat{\mathbf{m}}_t = \mathbf{m}(T_t(\hat{s}^{\mathrm{E}}_t))$;
    $\hat{s}^{\mathrm{E}}_{t-1} = s^{\mathrm{S}}(T_t(\hat{s}^{\mathrm{E}}_t))$;
end for

Algorithm 2 The Viterbi algorithm (VA) for hard-decision inputs using indexes on the right-hand side of the trellis.

Input: The received signal $\mathbf{r} = \{\mathbf{r}_0, \mathbf{r}_1, \cdots, \mathbf{r}_{L-1}\}$.
Input: A Boolean flag TERM to indicate if the trellis is terminated.
Input: Trellis look-up tables indexed by the right-hand edge index e^{R}: $s^{\mathrm{S}}(e^{\mathrm{R}})$, $\mathbf{m}(e^{\mathrm{R}})$, and $\mathbf{c}(e^{\mathrm{R}})$.
Output: The estimated symbol sequence $\hat{\mathbf{m}} = \{\hat{\mathbf{m}}_0, \hat{\mathbf{m}}_1, \cdots, \hat{\mathbf{m}}_{L-1}\}$.
Initialization:
if TERM == TRUE **then**,
    $A_{-1}(0) = 0$;
    $A_{-1}(s) = +\infty$ for $1 \le s \le N_{\mathrm{S}} - 1$;
else
    $A_{-1}(s) = 0$ for $0 \le s \le N_{\mathrm{S}} - 1$;
end if;
Main Algorithm:
// Forward recursion with metric and survivor updates
for $t = 0, 1, \cdots, L-1$ **do**
    for $s = 0, 1, \cdots, N_{\mathrm{S}} - 1$ **do**
        $A_t(s) = \min\limits_{2^k s \le e^{\mathrm{R}} \le 2^k(s+1)-1} \left\{A_{t-1}(s^{\mathrm{S}}(e^{\mathrm{R}})) + \gamma_t(e^{\mathrm{R}})\right\}$;
        $T_t(s) = \arg\min\limits_{2^k s \le e^{\mathrm{R}} \le 2^k(s+1)-1} \left\{A_{t-1}(s^{\mathrm{S}}(e^{\mathrm{R}})) + \gamma_t(e^{\mathrm{R}})\right\}$;
        where $\gamma_t(e^{\mathrm{R}}) = d_{\mathrm{H}}\big(\mathbf{r}_t, \mathbf{c}(e^{\mathrm{R}})\big)$;
    end for
end for
// Final global survivor at the end of the transmission
if TERM == TRUE **then**,
    $\hat{s}^{\mathrm{E}}_{L-1} = 0$;
else
    $\hat{s}^{\mathrm{E}}_{L-1} = \arg\min\limits_{0 \le s \le N_{\mathrm{S}}-1} \{A_{L-1}(s)\}$;
end if;
// Traceback operation to identify output sequence
for $t = L-1, \cdots, 1, 0$ **do**
    $\hat{\mathbf{m}}_t = \mathbf{m}(T_t(\hat{s}^{\mathrm{E}}_t))$;
    $\hat{s}^{\mathrm{E}}_{t-1} = s^{\mathrm{S}}(T_t(\hat{s}^{\mathrm{E}}_t))$;
end for

Algorithm 3 The Viterbi algorithm (VA) for soft-decision inputs using indexes on the right-hand side of the trellis.

Input: The received signal $\mathbf{r} = \{\mathbf{r}_0, \mathbf{r}_1, \cdots, \mathbf{r}_{L-1}\}$.
Input: A Boolean flag TERM to indicate if the trellis is terminated.
Input: Trellis look-up tables indexed by the right-hand edge index e^{R}: $s^{\mathrm{S}}(e^{\mathrm{R}})$, $\mathbf{m}(e^{\mathrm{R}})$, and $\mathbf{a}(e^{\mathrm{R}})$.
Output: The estimated symbol sequence $\hat{\mathbf{m}} = \{\hat{\mathbf{m}}_0, \hat{\mathbf{m}}_1, \cdots, \hat{\mathbf{m}}_{L-1}\}$.
Initialization:
if TERM == TRUE **then**,
    $A_{-1}(0) = 0$;
    $A_{-1}(s) = -\infty$ for $1 \le s \le N_{\mathrm{s}} - 1$;
else
    $A_{-1}(s) = 0$ for $0 \le s \le N_{\mathrm{s}} - 1$;
end if;
Main Algorithm:
// Forward recursion with metric and survivor updates
for $t = 0, 1, \cdots, L-1$ **do**
    for $s = 0, 1, \cdots, N_{\mathrm{s}} - 1$ **do**
        $A_t(s) = \max\limits_{2^k s \le e^{\mathrm{R}} \le 2^k(s+1)-1} \left\{A_{t-1}(s^{\mathrm{S}}(e^{\mathrm{R}})) + \gamma_t(e^{\mathrm{R}})\right\}$;
        $T_t(s) = \arg\max\limits_{2^k s \le e^{\mathrm{R}} \le 2^k(s+1)-1} \left\{A_{t-1}(s^{\mathrm{S}}(e^{\mathrm{R}})) + \gamma_t(e^{\mathrm{R}})\right\}$;
        where $\gamma_t(e^{\mathrm{R}}) = \sum_{i=0}^{n-1} r_t^{(i)} a^{(i)}(e^{\mathrm{R}}) = \mathbf{a}(e^{\mathrm{R}})\mathbf{r}_t^T$;
    end for
end for
// Final global survivor at the end of the transmission
if TERM == TRUE **then**,
    $\hat{s}_{L-1}^{\mathrm{E}} = 0$;
else
    $\hat{s}_{L-1}^{\mathrm{E}} = \arg\max\limits_{0 \le s \le N_{\mathrm{s}}-1} \{A_{L-1}(s)\}$;
end if;
// Traceback operation to identify output sequence
for $t = L-1, \cdots, 1, 0$ **do**
    $\hat{\mathbf{m}}_t = \mathbf{m}(T_t(\hat{s}_t^{\mathrm{E}}))$;
    $\hat{s}_{t-1}^{\mathrm{E}} = s^{\mathrm{S}}(T_t(\hat{s}_t^{\mathrm{E}}))$;
end for