Homework #3
EECS 869: Error Control Coding
Fall 2017

Consider the case where we have $n_0 = 3$, and thus $M_c = 2^{n_0-1} = 8$. At each time step t, the CC encoder outputs a binary 3-tuple $\mathbf{c}_t \triangleq \left[c_t^{(0)}, c_t^{(1)}, c_t^{(2)}\right]$, which can be represented by a decimal value $C_t \in \{0, 1, \ldots, 7\}$ using a standard binary-to-decimal conversion of $C_t = 4c_t^{(0)} + 2c_t^{(1)} + c_t^{(2)}$. The binary 3-tuple also has an antipodal representation given by $a(\mathbf{c}_t) = 2\mathbf{c}_t - 1$. We have a 3-tuple of log-likelihood ratios (LLRs) for $\mathbf{c}_t$ at time step t, $\boldsymbol{\lambda}_t(c) \triangleq [\lambda_{n_0t+0}(c), \lambda_{n_0t+1}(c), \lambda_{n_0t+2}(c)] = [0.5, -0.6, 0.1]$. Use MATLAB (or some other computer method) to make the following computations. Submit the numerical answers along with your code.

(1) Starting with the three LLRs, use the relationship between LLRs and log-domain binary pmfs, i.e. $p_{n_0t+j}(c = 1) = +\lambda_{n_0t+j}(c)/2$, $p_{n_0t+j}(c = 0) = -\lambda_{n_0t+j}(c)/2$, to produce three log-domain binary pmfs (six numerical values in total). Note: log-domain pmfs do not have any kind of "normalization."

(2) Take the log-domain binary pmfs you just obtained and convert them to linear-domain binary pmfs. The relationship between the two domains is $p_{n_0t+j}(c) = \ln(P_{n_0t+j}(c))$, although you are going the other direction (i.e., from log to linear). When the pmfs first arrive in the linear domain they will not sum to unity, so you must take the additional step to normalize them so you end up with proper pmfs.

(3) Now run the computations you did in Problems (1) and (2) in reverse so you end up with the same three LLR values you started with. You should observe that the values of the log-domain binary pmfs in the middle step will be different from the values you had before. This is because of the normalization step. We conclude that the normalization step is necessary to keep the values within a certain numerical range to prevent computer overflow/underflow; however, the particular normalization rule (i.e. pmfs must sum to unity) is arbitrary.

(4) Starting again with the three LLRs, use the relationship between LLRs and log-domain M_c-ary pmfs, i.e. $p_t(C) = \frac{1}{2}\boldsymbol{\lambda}_t(c)a(\mathbf{c})^T = \frac{1}{2}\sum_{j=0}^{n_0-1}\lambda_{n_0t+j}(c)a(c^{(j)})$, to produce the $M_c = 8$ numerical values of the log-domain pmf for C_t. Although there is no normalization for these values, identify the value of C_t with the largest pmf value and provide some explanation as to why this one is the largest.

(5) Take the log-domain M_c-ary pmf you just obtained and convert it to a linear-domain, normalized, M_c-ary pmf. The relationship between the two domains is the same as before, i.e. $p_{n_0t+j}(c) = \ln(P_{n_0t+j}(c))$. Confirm whether or not the value of C_t with the largest pmf value is the same a in Problem (4).

(6) Take the linear-domain M_c-ary pmf you just obtained and marginalize it three different times to obtain three binary pmfs (six numerical values in total). The resulting values should match the ones you obtained in Problem (2) and no re-normalization should be needed.

(7) Take the three binary pmfs and use them to reconstruct the M_c-ary pmf you had previously. The relationship needed for this is $P_t(C) = \prod_{j=0}^{n_0-1} P_{n_0t+j}(c^{(j)})$. Note the similarity between the log-domain computation $p_t(C) = \frac{1}{2}\boldsymbol{\lambda}_t(c)a(\mathbf{c})^T = \frac{1}{2}\sum_{j=0}^{n_0-1}\lambda_{n_0t+j}(c)a(c^{(j)})$ and the linear-domain computation that was just given.