Communication Systems Laboratory
Lab 5: Channel Coding

(Report Due: 23:00, November 24, 2021)

1 Overview

In Information and Communication Technology (and even Quantum Information Technology), error correction codes (ECC) are used for correcting (or detecting) errors in data over noisy communication channels or imperfect computing processes. The central idea is that the sender encodes the original messages which lives in a smaller dimensional space into a larger spaces so that the messages are separate as far as possible in the lager space by adding redundancy as in the form of an ECC.

In the lecture, we studied the fundamentals of convolutional codes, which can be well explained using finite-state machines, state diagrams, and trellis diagrams [1, 2]. With these concepts, convolutional encoding is related to **a path on the trellis diagram**, and the convolutional decoding can be viewed as **finding the closest path** on the trellis diagram according to the minimum Hamming distance rule[1]. This problem can be solved efficiently using the **Viterbi algorithm**. In this lab, we will implement convolutional encoders and decoders using `Matlab`.

2 Experiments

1. **(15 points) Analysis of Convolutional Codes:** In this problem, we consider a convolutional encoder with two finite impulse responses (FIR)

$$\mathbf{g}^{(1)} = \begin{bmatrix} 1 & 0 & 1 \end{bmatrix}, \qquad \mathbf{g}^{(2)} = \begin{bmatrix} 1 & 1 & 1 \end{bmatrix}. \tag{1}$$

   Do the following problems by your hand to make sure that you understand the mechanism of convolutional codes completely.

   (a) Calculate the code rate R.

   (b) Draw the shift register structure.

   (c) Draw the state transition diagram.

   (d) Draw the trellis diagram.

   (e) Illustrate the path on the trellis diagram for the received sequence

$$\boldsymbol{y} = \begin{bmatrix} 0 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \end{bmatrix}.$$

   Also list the decoded codewords $\widehat{\boldsymbol{c}} = \{\widehat{c}_i\}$ and the decoded bits $\widehat{\boldsymbol{b}} = \{\widehat{b}_i\}$.

   (f) Based on your results in Problem 1e, calculate the Hamming distance between $\boldsymbol{y}$ and $\widehat{\boldsymbol{c}}$.

[1]Note that the minimum maximum likelihood decoding rule is equivalent to the minimum Hamming distance rule *when the underlying channel* are memoryless binary symmetric channels.

2. **(60 points) Implementation of convolutional code #1:**

   (a) **(20 points)** Write a `Matlab` function to implement a convolutional encoder. The function should have the following arguments:

   ```
   encoded_data = conv_enc(binary_data, impulse_response);
   ```

   The input argument `binary_data` is the input binary sequence $\{b_i\}$ and `encoded_data` is the associated output binary sequence $\{c_i\}$.

   The argument `impulse_response` contains the impulse responses of the FIR filters. For example, the FIR filters used in the class are

   $$\mathbf{g}^{(1)} = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix}, \qquad \mathbf{g}^{(2)} = \begin{bmatrix} 1 & 0 & 1 \end{bmatrix}, \qquad \mathbf{g}^{(3)} = \begin{bmatrix} 1 & 1 & 1 \end{bmatrix}. \tag{2}$$

   This means that

   ```
   impulse_response = [1, 0, 0; 1, 0, 1; 1, 1, 1];
   ```

   Verify your results with the example:

   $$\text{Input bits } \mathbf{b} = \begin{bmatrix} 1 & 0 & 1 & 1 & 0 \end{bmatrix}$$

   to the associated output binary sequence $\mathbf{c} = \{c_i\}_i$ is the same as what taught in class.

   Hint: In your implementation, you may first build a table for the finite state machine of the convolutional code. Then use this table to encode the binary data.

   (b) **(20 points)** Implement a `Matlab` function for a convolutional decoder. The function has the following arguments:

   ```
   decoded_data = conv_dec(binary_data, impulse_response);
   ```

   Verify your results with the example :

   $$\boldsymbol{d} = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 \end{bmatrix}.$$

   and also verify the results in Problem 1e.

   (c) **(20 points)** Please do the following simulation.

   (i) Randomly generate a uniform (i.e. both 0 and 1 are equiprobable) bit sequence as the information bits with length more than 100000.

   (ii) Send it through your `Matlab` function `encoded_data` with the FIR given in (2).

   (iii) Apply a *memoryless binary symmetric channel*[2] where the coded bit 0 will transit to 1 with probability p and the coded bit 1 will also transit to 0 with probability p on the coded sequence obtained from your `encoded_data`.

   (iv) Decode the noise-corrupted sequence by your your `Matlab` function `decoded_data`.

[2]You can understand it as a random bit flipping process. Using the quantum language, you apply an X NOT Gate with probability p.

(v) Calculate the *bit error rate* (i.e. to count how many erroneous bits occurred divided by the length of the information bit sequence).

(vi) Repeat the above procedure for $p = 0, 0.1, 0.2, \dots, 1$. What did you observe when $p > 1/2$?

Plot the simulation results.

3. **(25 points) Implementation of convolutional code #2:** In this problem, we consider the convolutional codes with the FIR:

$$\mathbf{g}^{(1)} = \begin{bmatrix} 1 & 1 & 0 \end{bmatrix}, \qquad \mathbf{g}^{(2)} = \begin{bmatrix} 1 & 0 & 1 \end{bmatrix}. \tag{3}$$

Please do the simulation described in Problem 2c but with the FIR given in (3). How is the simulation results compared with that in Problem 2c? Why?

3 Lab Report

There is no format/typesetting requirements for your lab report, but you have to make your report decent and looking nice. In the report, you should address the results of the exercises mentioned above. You should also include your simulation program in the appendix of the report. Include whatever discussions about the new findings during the lab exercise, or the problems encountered and how are those solved. Please properly cite the literature if you referred to. Do not limit yourself to the exercises specified here. You are highly encouraged to play around with your simulation program on self-initiated extra lab exercises/discussions.

Ideally, your `Matlab` program should implement all (n, k)-convolutional codes with $k = 1$. (Hopefully, you have achieved this in this Lab.) More generally, a convolutional code might take $k > 1$ inputs at a time to produce the n-bit coded bits. Note that convolutional codes are very fundamental in numerous fields. You may extend your codes to the Turbo codes, LDPC codes, or moreover the trellis-coded modulation schemes. Your Viterbi algorithm also has applications in language processing and Markov decision processes. So, congratulations! You have done a great job in this Lab. Hope your codes can be used elsewhere at some point in the future.

References

[1] S. Haykin, *Communication Systems*, 4th ed. John Wiley & Sons, Inc., 2000.

[2] J. G. Proakis and M. Salehi, *Digital Communications*, 5th ed. McGraw-Hill, 2008.