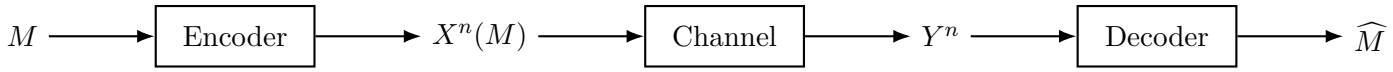
Introduction to Information and Coding Theory

Lecture 8: Convolutional Codes

Lecturer: Haim Permuter Scribe: Shay Shitrit

1 Channel coding recap

The goal in channel coding is to transmit a message through a noisy channel and recover it with arbitrarily small probability of error. The general setting is



A code of rate R maps information bits into a longer channel input sequence. The requirement is

$$P_e^{(n)} = \Pr\{\widehat{M} \neq M\} \xrightarrow[n\to\infty]{} 0.$$

In this lecture we focus on *convolutional codes*. The broader family of modern error correcting codes also includes turbo codes, LDPC codes and polar codes. Convolutional codes are important because they introduce the ideas of memory, state diagrams, trellises and maximum-likelihood sequence decoding.

2 Convolutional encoder

A convolutional encoder is a linear finite-state machine. Unlike a block code, the output at time t depends not only on the current input bit, but also on a finite number of previous input bits.

Definition 1 (Rate)**.** *If the encoder receives k information bits at each time step and outputs n coded bits, then the code rate is*

$$R = \frac{k}{n}.$$

Example 1 (Rate 1/2 convolutional encoder)**.** *Consider the encoder in Figure 1. At each time t, the encoder receives one information bit u_t and outputs two coded bits. Therefore $k = 1$, $n = 2$, and the rate is $R = 1/2$.*

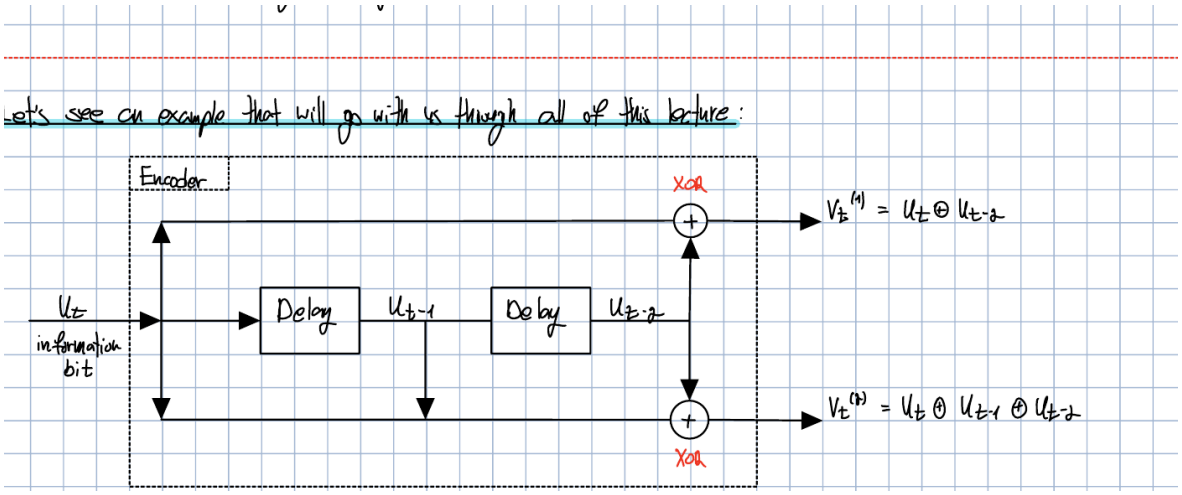


Figure 1: A rate 1/2 convolutional encoder with two memory elements. The plus signs are XOR operations.

The output equations are

$$v_t^{(1)} = u_t \oplus u_{t-2}, \tag{1}$$

$$v_t^{(2)} = u_t \oplus u_{t-1} \oplus u_{t-2}, \tag{2}$$

where $\oplus$ denotes XOR. The arithmetic is over $\mathbb{F}_2$; writing XOR emphasizes the actual circuit operation.

3 Generator sequences

The encoder can be described by generator sequences. For the example above,

$$g^{(1)} = (1, 0, 1), \qquad g^{(2)} = (1, 1, 1).$$

The general form is a binary convolution over $\mathbb{F}_2$:

$$v_t^{(j)} = \sum_{\ell=0}^{m} u_{t-\ell} \cdot g_\ell^{(j)} \quad \text{over } \mathbb{F}_2,$$

where m is the memory of the encoder. Equivalently,

$$v^{(j)} = u * g^{(j)} \quad \text{over } \mathbb{F}_2.$$

For this encoder, $m = 2$. The state at time t is chosen as

$$s_t = (u_{t-1}, u_{t-2}).$$

Thus there are $2^m = 4$ possible states: $00, 01, 10, 11$.

4 Encoding example

Assume the initial state is zero, namely $u_0 = 0$ and $u_{-1} = 0$, and let

$$u = [u_1, u_2, \ldots, u_7] = [1, 0, 1, 1, 1, 0, 0].$$

For the first output stream, using $v_i^{(1)} = u_i \oplus u_{i-2}$:

$$\begin{aligned}
v_1^{(1)} &= u_1 \oplus u_{-1} = 1 \oplus 0 = 1, \\
v_2^{(1)} &= u_2 \oplus u_0 = 0 \oplus 0 = 0, \\
v_3^{(1)} &= u_3 \oplus u_1 = 1 \oplus 1 = 0, \\
v_4^{(1)} &= u_4 \oplus u_2 = 1 \oplus 0 = 1, \\
v_5^{(1)} &= u_5 \oplus u_3 = 1 \oplus 1 = 0, \\
v_6^{(1)} &= u_6 \oplus u_4 = 0 \oplus 1 = 1, \\
v_7^{(1)} &= u_7 \oplus u_5 = 0 \oplus 1 = 1.
\end{aligned}$$

Therefore,

$$v^{(1)} = [1, 0, 0, 1, 0, 1, 1].$$

For the second output stream, using $v_i^{(2)} = u_i \oplus u_{i-1} \oplus u_{i-2}$:

$$\begin{aligned}
v_1^{(2)} &= u_1 \oplus u_0 \oplus u_{-1} = 1 \oplus 0 \oplus 0 = 1, \\
v_2^{(2)} &= u_2 \oplus u_1 \oplus u_0 = 0 \oplus 1 \oplus 0 = 1, \\
v_3^{(2)} &= u_3 \oplus u_2 \oplus u_1 = 1 \oplus 0 \oplus 1 = 0, \\
v_4^{(2)} &= u_4 \oplus u_3 \oplus u_2 = 1 \oplus 1 \oplus 0 = 0, \\
v_5^{(2)} &= u_5 \oplus u_4 \oplus u_3 = 1 \oplus 1 \oplus 1 = 1, \\
v_6^{(2)} &= u_6 \oplus u_5 \oplus u_4 = 0 \oplus 1 \oplus 1 = 0, \\
v_7^{(2)} &= u_7 \oplus u_6 \oplus u_5 = 0 \oplus 0 \oplus 1 = 1.
\end{aligned}$$

Therefore,

$$v^{(2)} = [1, 1, 0, 0, 1, 0, 1].$$

The channel input is the serialized sequence

$$(v_1^{(1)}, v_1^{(2)}, v_2^{(1)}, v_2^{(2)}, \ldots, v_7^{(1)}, v_7^{(2)}) = [1, 1, 0, 1, 0, 0, 1, 0, 0, 1, 1, 0, 1, 1].$$

5 State transition table

Given the current state $s_t = (u_{t-1}, u_{t-2})$ and input u_t, the next state is

$$s_{t+1} = (u_t, u_{t-1}).$$

The output is $(v_t^{(1)}, v_t^{(2)})$.

Input u_t	Current state (u_{t-1}, u_{t-2})	Next state (u_t, u_{t-1})	Output
0	00	00	00
1	00	10	11
0	01	00	11
1	01	10	00
0	10	01	01
1	10	11	10
0	11	01	10
1	11	11	01

A transition label is written as u_t/v_t, meaning input/output.

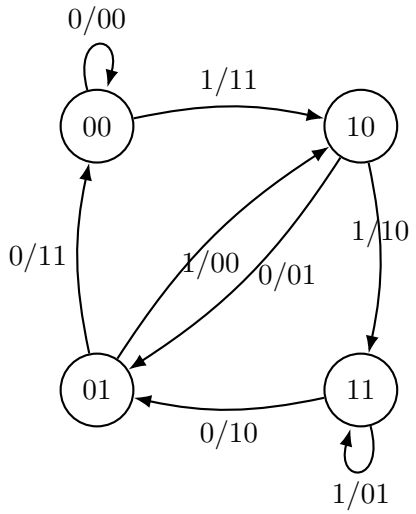


Figure 2: State diagram of the convolutional encoder.

6 Trellis representation

The state diagram is unfolded in time to form a trellis. Each column represents a time step. Each node represents a possible state, and each branch represents a legal transition from one state to the next. Therefore, each path through the trellis corresponds to one possible input sequence u and one possible codeword sequence v.

The notes use the following notation:

$$u = [u_0, u_1, \ldots, u_{L-1}],$$

$$v = [v_1, \ldots, v_N, \ldots, v_{N+m-1}], \qquad N = n \cdot L,$$

where L is the number of input time steps, n is the number of output bits per step, and m is the memory, i.e. the number of delays. The received sequence is denoted by

$$r = [r_1, \ldots, r_N, \ldots, r_{N+m-1}].$$

The trellis and the channel model used in the notes are shown in Figure 3. The small trellis diagram shows the transition from time t to time $t+1$. The right-hand diagram is a binary symmetric channel (BSC), in which the transmitted bit is flipped with probability p and is kept unchanged with probability $1-p$.

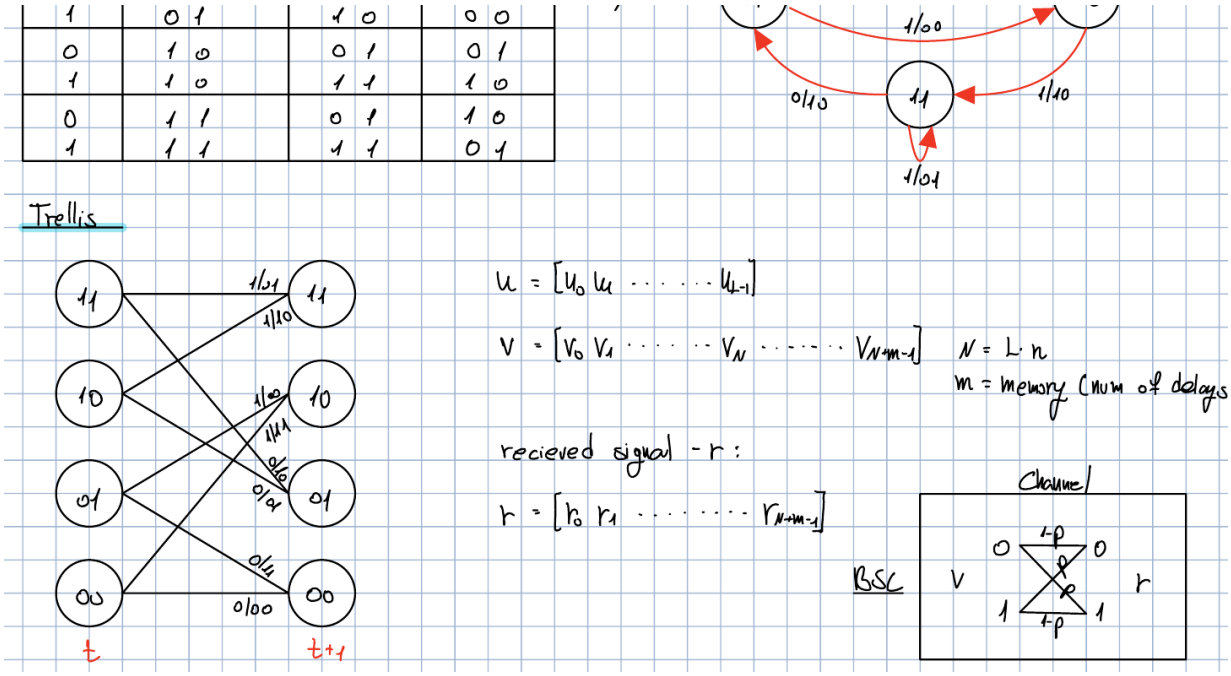


Figure 3: Trellis notation and BSC diagram from the handwritten notes.

7 Maximum-likelihood decoding

For a memoryless channel, the likelihood of a received sequence r given a candidate codeword v factors as

$$P(r|v) = \prod_{t=0}^{N+m-1} P(r_t|v_t).$$

Thus the maximum-likelihood decoder chooses

$$\widehat{v} = \arg\max_{v} P(r|v) = \arg\max_{v} \prod_{t=0}^{N+m-1} P(r_t|v_t).$$

When we find $\widehat{v}$, we can recover the corresponding input sequence $\widehat{u}$ from the trellis path.

Taking logarithms gives

$$\widehat{v} = \arg\max_{v} \log P(r|v) = \arg\max_{v} \sum_{t=0}^{N+m-1} \log P(r_t|v_t).$$

Equivalently, this can be written as a path-metric problem:

$$\widehat{v} = \arg\max_{v} \sum_{t=0}^{N+m-1} M(r_t|v_t).$$

For the BSC, maximizing the likelihood is equivalent to minimizing the Hamming distance between the received sequence and the candidate codeword:

$$\widehat{v} = \arg\min_{v} d_H(r, v).$$

Definition 2 (Hamming distance)**.** *The Hamming distance between two binary sequences of the same length is the number of positions in which they differ:*

$$d_H(x, y) = \sum_i \mathbf{1}\{x_i \neq y_i\}.$$

For example,

$$s_1 = 0101, \qquad s_2 = 0010,$$

then

$$s_1 \oplus s_2 = 0111,$$

so

$$d_H(s_1, s_2) = 3.$$

8 Viterbi decoding

The Viterbi algorithm is a dynamic-programming algorithm for finding the shortest path in the trellis. Instead of checking all possible paths, it keeps only the best path that reaches each state at each time. This remaining path is called the *survivor path.*

Example 2 (Viterbi decoding)**.** *Suppose the received sequence is grouped into pairs as*

$$r = (01, 11, 10, 10, 00, 11, 10).$$

We want to find the most likely codeword v, and then obtain the corresponding input sequence u. We build the trellis through time and compare each branch output with the received pair at the same time.

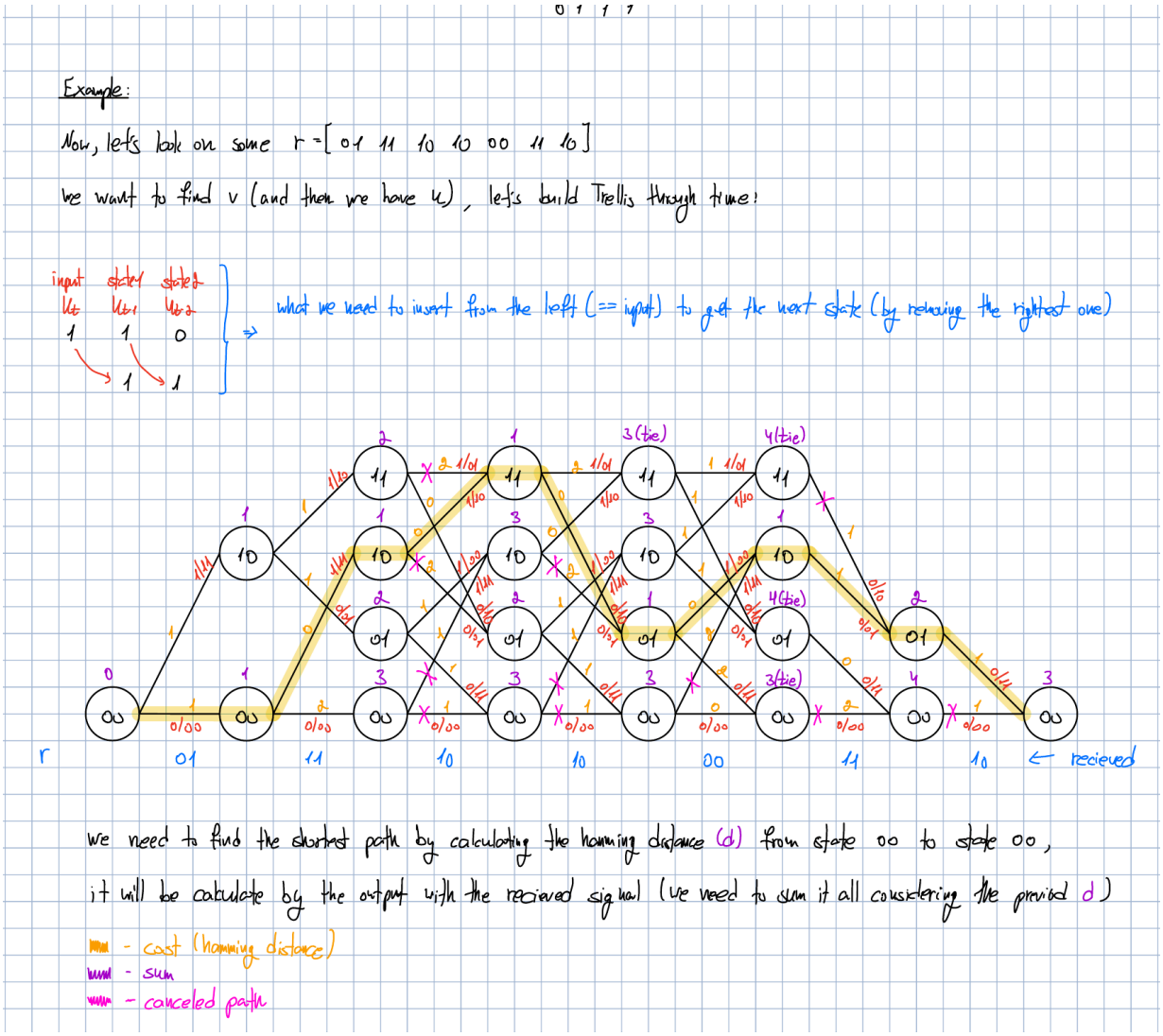


Figure 4: Viterbi decoding example from the notes.

The explanation of the diagram is:

- *The received sequence appears under the trellis in blue:* $01, 11, 10, 10, 00, 11, 10$.
- *For each branch, compare the branch output with the received pair and compute the Hamming distance.*
- *The branch costs are added along the path, so the path metric is a cumulative sum.*
- *When two paths enter the same state, keep only the one with the smaller accumulated metric and cancel the other path.*
- *The goal in the drawn example is to find the shortest path from state* 00 *to state* 00.

The color mapping in the handwritten diagram is:

Color	*Meaning*
Orange	*Branch cost, i.e. Hamming distance*
Purple	*Accumulated sum / path metric*
Pink	*Canceled path*
Yellow highlight	*Survivor path*

The branch cost at time t is

$$cost = d_H(r_t, branch\ output).$$

The selected path is the one with minimum accumulated Hamming distance.

9 Summary

- A convolutional code has memory: each output depends on the current input and previous inputs.
- The encoder is described by generator sequences $g^{(j)}$.
- The state is the content of the memory elements.
- The state diagram summarizes all possible transitions.
- The trellis is the time-unfolded state diagram.
- For a BSC, maximum-likelihood decoding is equivalent to minimizing Hamming distance.
- The Viterbi algorithm efficiently finds the minimum-metric path through the trellis.