Convolutional Code Decoder Documentation

Ananth Madhav V.

April 19, 2025

1 Functions

1.1 build_trellis(generator_polys: List[int], constraint_length: int) → Dict[tuple, tuple]

Purpose: Constructs trellis diagram for convolutional encoding.

Methodology:

- Generates 2^{K-1} states where K = constraint length
- For each state and input bit (0/1):
  1. Computes next state using bit shift: $\text{next\_state} = (\text{state} \ll 1) \& (\text{n\_states} - 1)$
  2. Calculates output bits via polynomial XOR operations:

$$\text{output\_bits}[j] = \bigoplus_{i=0}^{K-1} (g_j^{(i)} \cdot \text{state\_bit}[i])$$

  where g_j is the j-th generator polynomial

Key Parameters:

- `generator_polys`: Octal representations of generator polynomials
- `constraint_length`: Determines state space size and memory depth

1.2 viterbi_decode(received: List[int], trellis: Dict[tuple, tuple], n_states: int, n_outputs: int) → List[int]

Purpose: Implements maximum likelihood sequence estimation using Viterbi algorithm.

Methodology:

1. Initializes path metrics matrix with ∞ except start state (0)
2. Forward pass:
   - For each timestep, calculates Hamming distance between received bits and expected outputs
   - Updates path metrics using:

$$\text{path\_metric}[t, s] = \min(\text{path\_metric}[t-1, s'] + \text{branch\_metric})$$

   - Stores survivor paths
3. Traceback:
   - Starts from minimum metric state at final timestep
   - Reconstructs input bits by following survivor paths backward

2 Core Concepts

2.1 Convolutional Encoding

- Defined by (n, k, K) parameters:
  - $n = 2$: Number of output bits per input bit
  - $k = 1$: Number of input bits
  - $K = 3$: Constraint length (encoder memory)
- Generator polynomials determine output calculation:

$$G = [g^{(0)}(D), g^{(1)}(D)] = [D^2 + D + 1, D^2 + 1]$$

2.2 Trellis Representation

- Nodes represent encoder states (shift register contents)
- Edges show state transitions with input/output labels
- Example transition:

$$(s_t, u_t) \to s_{t+1} = (s_t \ll 1)|u_t,\ y_t = [y_t^{(0)}, y_t^{(1)}]$$

2.3 Viterbi Algorithm

- Optimal decoding for memoryless channels
- Key components:
  1. Branch metric: Hamming distance

$$\gamma_t(s', s) = \sum_{i=1}^{n} |r_t^{(i)} - y_t^{(i)}|$$

  2. Path metric update:

$$\Gamma_t(s) = \min_{s'}(\Gamma_{t-1}(s') + \gamma_t(s', s))$$

- Complexity: $O(N \cdot 2^K)$ for sequence length N

3 Implementation Notes

- Polynomial handling: Decimal inputs converted to binary coefficients
- State management: Bitmasking used for efficient state transitions
- Received sequence processing: Assumes hard-decision inputs
- Traceback: Implements register-exchange method with reverse traversal

4 Example Usage

```
Generator polynomials (decimal): 7,5 → octal 111,101
Constraint length: 3
Received sequence: 11 01 11 01 10
Decoded message: [1, 0, 1, 1, 0]
```

Parameter	Value
Constraint Length (K)	3
Memory Elements	2
States	4
Code Rate	1/2

5 Performance Characteristics

- Time Complexity: $O(N \times 2^{K-1})$
- Memory Complexity: $O(N \times 2^{K-1})$
- Optimality: Maximum likelihood sequence estimation

6 References

1. Viterbi, A. (1967). Error Bounds for Convolutional Codes
2. Forney, G.D. (1973). The Viterbi Algorithm
3. Johannesson & Zigangirov (2015). Fundamentals of Convolutional Coding
4. Code credit: R1 1776 on Perplexity.AI