Convolutional encoding

Finite State Machine

Channel models

The Viterbi algorithm

Coding and decoding with convolutional codes.
The Viterbi Algorithm.

Numerical Transmission Systems - CSE 3A

Grenoble INP - PHELMA - Ensimag

Outline of this section

1 Convolutional encoding
   - Principles
     - 1st point of view: infinite length block code
     - 2nd point of view: convolutions
     - Some examples

Block codes: main ideas

Repetition code

- TX: CODING THEORY
- RX: CPDING TOEORY

No way to recover from transmission errors, we need to add some redundancy at the transmitter side. Repetition of transmitted symbols make detection and correction possible:

- TX:CCC OOO DDD III NNN GGG TTT HHH EEE OOO RRR YYY
- RX:CCC OPO DDD III NNN GGD TTT OHO EEE OOO RRR YYY
- C O D I N G T O E O R Y: 2 corrections - 1 detection.

Beyond repetition ...
Better codes exist.
Convolutional encoding

Block codes: main ideas

Geometric view

k=1 bit of information
2 code words (length n=3): (000) (111)

RX: how can the receiver decide about transmitted words:
- (001), (010), (100): Detection + correction (000)
- (110), (101), (011): Detection + correction (111)
- (000) (111) (Probably) right

Linear block codes, e.g., Hamming codes.
- A binary linear block code takes \( k \) information bits at its input and calculates \( n \) bits. If the \( 2^k \) codewords are enough and well spaced in the \( n \)-dim space, it is possible to detect or even correct errors.
- In 1950, Hamming introduced the (7,4) Hamming code. It encodes 4 data bits into 7 bits by adding three parity bits.
- It can detect and correct single-bit errors but can only detect double-bit errors.
- The code parity-check matrix is:

\[
H = \begin{pmatrix}
1 & 0 & 1 & 0 & 1 & 0 & 1 \\
0 & 1 & 1 & 0 & 0 & 1 & 1 \\
0 & 0 & 0 & 1 & 1 & 1 & 1
\end{pmatrix}
\]

Convolutional encoding: main ideas

- In convolutional codes, each block of \( k \) bits is mapped into a block of \( n \) bits **BUT**
- these \( n \) bits are not only determined by the present \( k \) information bits but also by the previous information bits. This dependence can be captured by a finite state machine.
- This is achieved using several linear filtering operations:
  - Each convolution imposes a constraint between bits.
  - Several convolutions introduce the redundancy.

Outline of this section

1. Convolutional encoding
   - Principles
   - 1st point of view: infinite length block code
   - 2nd point of view: convolutions
   - Some examples
A convolutional code can be described by an "infinite matrix":

\[
G = \begin{pmatrix}
G_0 & G_1 & \cdots & G_M & 0_{k \times n} & \cdots \\
0_{k \times n} & G_0 & \cdots & G_{M-1} & G_M & \cdots \\
\vdots & \vdots & \ddots & \vdots & \vdots & \ddots \\
\vdots & \vdots & \ddots & 0_{k \times n} & G_0 & \cdots \\
\vdots & \vdots & \ddots & G_0 & G_1 & \cdots \\
\vdots & \vdots & \ddots & \vdots & 0_{k \times n} & \ddots \\
\vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \ddots \\
\end{pmatrix}
\]

This matrix depends on \( K = M + 1 \) \( k \times n \) sub-matrices \( \{G_i\}_{i=0,\ldots,M} \). \( K \) is known as the constraint length of the code.

Denoting by:
- \( I_j = (l_j \cdots l_{jk}) \) the \( j^{th} \) block of \( k \) informative bits,
- \( C_j = (C_{j1} \cdots C_{jn}) \) a block of \( n \) coded bits at the output.

Coding an infinite sequence of blocks (length \( k \)) \( I = (I_0 I_1 \cdots) \) produces an infinite sequence \( C = (C_0 C_1 \cdots) \) of coded blocks (length \( n \)).

Block form of the coding scheme: it looks like a block coding:

\[
C = IG
\]

Outline of this section

1. Convolutional encoding
   - Principles
   - 1st point of view: infinite length block code
   - 2nd point of view: convolutions
   - Some examples
Infinite generator matrix performs a convolution

Using the convention $I_i = 0$ for $i < 0$, the encoding structure $C = IG$ is clearly a convolution:

\[ C_j = \sum_{l=0}^{M} I_{j-l} G_{l}. \]

For an informative bits sequence $I$ whose length is finite, only $L < +\infty$ blocks of $k$ bits are different from zero at the input of the coder: $I = (I_0 \cdots I_{L-1})$. The sequence $C = (C_0 \cdots C_{L+M})$ at the coder output is finite too.

This truncated coded sequence is generated by a linear block code whose generator matrix is a size $kL \times n(L + M)$ sub-matrix of $G$.

Shift registers based realization

Let us write $g_{\alpha\beta}^{(l)}$ elements of matrix $G_l$. We now expand the convolution $C_j = \sum_{l=0}^{M} I_{j-l} G_{l}$ to explicit the $n$ components $C_{j1}, \ldots, C_{jn}$ of each output block $C_j$:

\[ C_j = [C_{j1}, \ldots, C_{jn}] = [\sum_{l=0}^{M} \sum_{\alpha=1}^{k} I_{j-l,\alpha} g_{\alpha\beta}^{(l)}, \ldots, \sum_{l=0}^{M} \sum_{\alpha=1}^{k} I_{j-l,\alpha} g_{\alpha\beta}^{(l)}] \]

If the length of the shift register is $L$, there are $M^L$ different internal configurations. The behavior of the convolutional coder can be captured by a $M^L$ states machine.

Rate of a convolutional code

Asymptotic rate

For each $k$ bits long block at the input, a $n$ bits long block is generated at the output. At the coder output, the ratio [number of informative bits] over [total number of bits] is given by:

\[ R = \frac{k}{n} \]

This quantity is called the rate of the code.
Rate of a convolutional code

Finite length rate
For a finite length input sequence, the truncating reduces the rate. The exact finite-length rate is exactly:

\[ r' = r \frac{L}{L + M} \]

For \( L \gg M \), this rate is almost equal to the asymptotic rate.

Outline of this section

1. Convolutional encoding
   - Principles
   - 1st point of view: infinite length block code
   - 2nd point of view: convolutions
   - Some examples

Shift registers based realization

Rate 1/2 encoder. 1 input \((k = 1)\), 2 outputs \((n = 2)\).

- 2 convolutions are evaluated in parallel.
- The output of each convolution depends on one input and on 3 values memorized in the shift register.
- At each step, the 2 values at the output depend on the input and the internal state.

Impulse responses are \( P(D) = 1 + D + D^2 + D^3 \) and \( Q(D) = 1 + D + D^2 \).

Rate 1/2 \((k = 1, n = 2)\)
- Constraint length \( K = M + 1 = 4 \)
Convolutional encoding

Finite State Machine Channel models

The Viterbi algorithm

Shift registers based realization

Rate 2/3 encoder

- 3 convolutions are evaluated in parallel.
- The output of each convolution depends on two inputs and on 4 values memorized in the shift registers.
- At each step, the 3 values at the output depend on the inputs and the internal state.

\[
\begin{align*}
I_j &= I_j^1, I_j^2, I_j^1, \\
G_0 &= \begin{pmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix}, & G_1 &= \begin{pmatrix} 0 & 0 & 1 \\ 0 & 0 & 1 \end{pmatrix}
\end{align*}
\]

Rate 2/3 \((k = 2, n = 3)\)
- Constraint length \(K = 2\)
- Sub-matrices: \(G_0 = \begin{pmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix}\) and \(G_1 = \begin{pmatrix} 0 & 0 & 1 \\ 0 & 0 & 1 \end{pmatrix}\)

Shift registers based realization

Rate 1/2 encoder

- 2 convolutions are evaluated in parallel.
- The output of each convolution depends on one input \(I_{j,1}\) and on 2 values memorized in the shift register \(I_{j-1,1}\) and \(I_{j-2,1}\).
- At each step, the 2 values at the output depend on the input and the internal state.

\[
\begin{align*}
I_j &= I_j^1, I_j^2, I_j^1, \\
G_0 &= \begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix}
\end{align*}
\]

Rate 1/2 \((k = 1, n = 2)\)
- Constraint length \(K = M + 1 = 3\)
- Sub-matrices: \(G_0 = \begin{pmatrix} 1 & 1 \end{pmatrix}, G_1 = \begin{pmatrix} 0 & 1 \end{pmatrix}\) and \(G_2 = \begin{pmatrix} 1 & 1 \end{pmatrix}\).
Shift registers based realization

Rate 1/2 encoder

This rate 1/2 \((k = 1, n = 2)\) code is used in the sequel to explain the Viterbi algorithm.

Outline

1. Convolutional encoding
2. Finite State Machine
3. Channel models
4. The Viterbi algorithm

Outline of this section

2. Finite State Machine
   - Transition diagram
   - Lattice diagram
   - A convolutional encoder is a FSM
   - Coding a sequence using the coder FSM

Finite State Machine

- A state space
- An input that activates the transition from one state to another one. An output is generated during the transition.
- Usual representations:
  - Transition diagram
  - Lattice diagram
Finite State Machine: transition diagram

- The state space is composed of 4 elements: 00, 01, 11, 10.
- Each state is represented by a node.
- The input is binary valued: 2 arrows start at each node.
- Arrows are indexed by a couple of values: (input, output) the input that activates the transition and the output that is generated by this transition.

Finite State Machine: lattice diagram

- A new input triggers a transition from the present state to a one-step future step.
- A lattice diagram unwraps the behavior of the FSM as a function of time.
A simple encoder ...

... and the FSM of this encoder

Coding a sequence using the coder FSM

Information bearing bits enter the coder

The first value at the coder input is: $I_0 = 0$.
According to the transition diagram, this input activates the transition indexed by $(0, 00)$: this means input 0 generates output $C_0 = (00)$. The transition is from state 00 to state 00.

The second value at the coder input is: $I_1 = 0$.
According to the transition diagram, this input activates the transition indexed by $(0, 00)$: this means input 0 generates output $C_1 = (00)$. The transition is from state 00 to state 00.
Coding a sequence using the coder FSM

Coding bits 001 with a known automate.

- Initial coder state: 00.
- Informative sequence: 001.

Information bearing bits enter the coder

The last informative bit to enter the coder is 1. According to the transition diagram, this input activates the transition indexed by (1, 11): this means input 1 generates output $C_2 = (11)$. The transition is from state 00 to state 10.

Lattice closure: 2 zeros at the coder input to reset its state.

- The first 0 activates the transition indexed by (0, 01), generates output $C_3 = (01)$ and sets the state to 01.
- The second 0 activates the transition indexed by (0, 11), generates output $C_4 = (11)$ and resets the state to 00.

Received sequence

In fine, the informative sequence 001 is encoded by:

$$[C_0, C_1, C_2, C_3, C_4] = [00, 00, 11, 01, 11]$$

Noisy received coded sequence

The coded sequence

$$[C_0, C_1, C_2, C_3, C_4] = [00, 00, 11, 01, 11]$$

is transmitted over a Binary Symmetric Channel. Let us assume that two errors occur and that the received sequence is:

$$[y_0, y_1, y_2, y_3, y_4] = [10, 01, 11, 01, 11]$$
Outline

1. Convolutional encoding
2. Finite State Machine
3. Channel models
4. The Viterbi algorithm

Outline of this section

3. Channel models
   - Binary Symmetric Channel
   - Additive White Gaussian Noise Channel

Binary Symmetric Channel

Diagram of the BSC

Characteristics of a Binary Symmetric Channel

- Memoryless channel: the output only depends on the present input (no internal state).
- Two possible inputs (0, 1), two possible outputs (0, 1),
- 0, 1 are equally affected by errors (error probability $p$)

Branch Metric of a Binary Symmetric Channel

Calculation of the Branch Metric

- The transition probabilities are:
  - $p(1|0) = p(0|1) = p$
  - $p(1|1) = p(0|0) = 1 - p$
- The Hamming distance between the received value $y$ and the coder output $C_{rs}$ (generated by the transition from state $r$ to state $s$) is the number of bits that differs between the two vectors.
Branch Metric of a Binary Symmetric Channel

Calculation of the Branch Metric

- The Likelihood is written:
  \[ p(y|C_{rs}) = (1 - p)^n \left( \frac{p}{1-p} \right)^{d_H(y, C_{rs})} \]
  \[ \log p(y|C_{rs}) = d_H(y, C_{rs}) \log \left( \frac{p}{1-p} \right) + n \log (1 - p) \]

- \( n \log (1 - p) \) is a constant and \( \log \left( \frac{p}{1-p} \right) < 0 \): maximizing the likelihood is equivalent to minimizing \( d_H(y, C_{rs}) \).
- The Hamming branch metric \( d_H(y, C_{rs}) \) between the observation and the output of the FSM is adapted to the BS Channel.

Additive White Gaussian Noise Channel

Diagram of the AWGN channel

\[ a_k \rightarrow n_k \rightarrow a_k + n_k \]

Characteristics of an AWGN channel

- Memoryless channel: the output only depends on the present input (no internal state).
- Two possible inputs \((-1, +1)\), real (or even complex)-valued output.
- The output is a superposition of the input and a Gaussian noise. 0, 1 are equally affected by errors.

Branch Metric of an AWGN Channel

Calculation of the Branch Metric

- The probability density function of a Gaussian noise is given by:
  \[ p(x) = \frac{1}{\sigma \sqrt{2\pi}} \exp \left( -\frac{x^2}{2\sigma^2} \right) \]

- The Euclidian distance between the analog received value \( y \) and the coder output \( C_{rs} \) (generated by the transition from state \( r \) to state \( s \)) is the sum of square errors between the two vectors.
**Branch Metric of an AWGN Channel**

**Calculation of the Branch Metric**

- The Likelihood is written:

\[
p(y|C_{rs}) = \left[ \frac{1}{\sigma \sqrt{2\pi}} \right]^n \exp \left( -\frac{|y - C_{rs}|^2}{2\sigma^2} \right)
\]

\[
\propto \exp \left( -\frac{d_E^2(y, C_{rs})}{2\sigma^2} \right)
\]

- Since

\[
\log p(y|C_{rs}) \propto -\frac{d_E^2(y, C_{rs})}{2\sigma^2}
\]

maximizing the likelihood is equivalent to minimizing the Euclidian distance.

---

**Outline**

1. Convolutional encoding
2. Finite State Machine
3. Channel models
4. The Viterbi algorithm

---

**Outline of this section**

4. The Viterbi algorithm
   - Problem statement
     - Main ideas of the Viterbi algorithm
     - Notations
     - Running a Viterbi algorithm

---

**Branch Metric of an AWGN Channel**

**Binary Symmetric Channel as an approximation of the Gaussian channel**

Note the BS channel is a coarse approximation of the AWGN channel with an error probability \( p \) given by:

\[
p = \frac{1}{\sigma \sqrt{2\pi}} \int_{1}^{+\infty} \exp \left( -\frac{x^2}{2\sigma^2} \right) dx
\]
Problem statement
- A finite-length (length $L$) sequence drawn from a finite-alphabet (size 2) enters a FSM.
- The FSM output goes through a channel that corrupts the signal with some kind of noise.
- The noisy channel output is observed

ML estimation of the transmitted sequence
- How can we find the input sequence that maximizes the likelihood of the observations?
  - Test $2^L$ input sequences!
  - Use the Viterbi algorithm to determine this sequence with a minimal complexity.

Outline of this section
- The Viterbi algorithm
  - Problem statement
  - Main ideas of the Viterbi algorithm
  - Notations
  - Running a Viterbi algorithm
Maximum Likelihood Sequence Estimation (MLSE)

**Likelihood**

The informative sequence is made of $L$ $k$-bits long blocks $I_j$:

\[ I = [I_0, \cdots, I_{L-1}] \]

This sequence is completed by $M$ blocks of $k$ zeros to close the lattice. The coded sequence is:

\[ C = [C_0, \cdots, C_{L-M}] \]

For a memoryless channel, the received sequence is:

\[ y = [y_0, \cdots, y_{L-M}] \]

Since observation $y_j$ depends on $s_j$ and $C_j$ or equivalently $s_js_{j+1}$, the likelihood function is given by:

\[
\log P(y|C) = \sum_{j=0}^{L-M-1} \log P(y_j|s_j, C_j) = \sum_{j=0}^{L-M-1} \log P(y_j|s_js_{j+1})
\]

- $l_j(s_j, s_{j+1}) = \log P(y_j|s_js_{j+1})$ is called a branch metric.
- Sums of branch metrics $\sum_{j=0}^k l_j(s_j, s_{j+1})$ are called path (or cumulative) metrics (from the initial node to $s_{j+1}$).

Searching for the optimal input sequence is equivalent to finding the path in the lattice whose final cumulative metric is maximal.
Notations for transitions on the lattice diagram

- **Previous state.** The weight of the optimal path that reaches this node is \( q \).
- **Branch metric.** Weight \( p \) and state \( s \) of the optimal path at this node.
- **Cumulative (path) metric.** \( q + b \).
- **Edge indexed by the couple (input, output).**

The weight of the optimal path that reaches this node is \( q \).

Viterbi for decoding the output of a BSC

Viterbi in progress ... open the lattice (1/2)

- The FSM memorizes two past values: it takes two steps to reach the steady-state behavior.
- During the first two steps, only a fraction of the impulse response is used. The lattice is being opened during this transient period.
- The initial state is 00, since the input is binary-valued, two edges start from state 00 and only 2 states are reachable: 00 and 10.

Computation of branch metrics:
- Branch 00 → 00: the FSM generates the output 00, the Hamming distance between this FSM output and the channel output 10 is 1.
- Branch 00 → 10: the FSM generates the output 11, the Hamming distance between this FSM output and the channel output 10 is 1.
- Each branch metric is surrounded by a circle on the figure.
Viterbi for decoding the output of a BSC

Viterbi in progress ... open the lattice (1/2)

Computation of cumulative (or path) metrics:
- A cumulative metric is the sum of branch metrics to reach a given node. Its value is printed on the edge just before the node.
- The path metric of the best path that reaches a node is printed in the black square representing this node. The initial path metrics is set to 0.

10 01 11 01 11
1
1
(1,11)
1
0
1
2

00 01 10 11
1
(0, 00)
1
0
2
(0, 01)
3
1
2
(1,10)
0
2
2
(1, 11)
1
0
1
2
3
1
2

Viterbi for decoding the output of a BSC

Viterbi in progress ... open the lattice (2/2)

Computations of cumulative (or path) metrics:
- Branch 00 → 00: the path metric to reach state 00 is equal to the previous path metric (0 for the initial path metric) plus the branch metric 1: path metric = 1.
- Branch 00 → 10: the path metric to reach state 10 is equal to the previous path metric (0 for the initial path metric) plus the branch metric 1: path metric = 1.

10 01 11 01 11
1
1
(1,11)
1
0
1
2

00 01 10 11
1
(0, 00)
1
0
2
(0, 01)
3
1
2
(1,10)
0
2
2
(1, 11)
1
0
1
2
3
1
2

Viterbi for decoding the output of a BSC

Viterbi in progress ... steady-state behavior

For each arrival node, the algorithm must select the path whose metric is minimal.
The Viterbi algorithm

Viterbi for decoding the output of a BSC

Viterbi in progress ... close the lattice (1/2)

The informative sequence is finished. Two zero inputs reset the state to its initial value.

(VHELMA/Ensimag/CSE 3A)