Performance Measures for Convolutional Codes

Whereas there is one main performance measure for block codes, minimum distance, there are several possible performance measures for convolutional codes. The following three are considered here:

1. The column distance function.
2. Minimum distance.
3. Minimum free distance.

The Column Distance Function (1)

Consider a rate-k/n code C with constraint length K. Let the input sequence $\mathbf{x}$ and the output sequence $\mathbf{y}$ for an encoder for C be truncated at length i:

$$[\mathbf{x}]_i = (x_0^{(0)}, \ldots, x_0^{(k-1)}, x_1^{(0)}, \ldots, x_1^{(k-1)}, \ldots, x_{i-1}^{(0)}, \ldots, x_{i-1}^{(k-1)}),$$
$$[\mathbf{y}]_i = (y_0^{(0)}, \ldots, y_0^{(n-1)}, y_1^{(0)}, \ldots, y_1^{(n-1)}, \ldots, y_{i-1}^{(0)}, \ldots, y_{i-1}^{(n-1)}).$$

The **column distance function** (CDF) d_i is the minimum Hamming distance between all pairs of output sequences truncated at length i given that the input sequences differ in the first k bits:

$$d_i := \min\{d([\mathbf{y}']_i, [\mathbf{y}'']_i) \mid [\mathbf{x}']_1 \neq [\mathbf{x}'']_1\}.$$

The Column Distance Function (2)

If the convolutional code is linear, the CDF can be defined in the following way:

$$d_i := \min\{w([\mathbf{y}]_i) \mid [\mathbf{x}]_1 \neq \mathbf{0}\}.$$

Example. The CDF for the code in [Wic, Fig. 11-1] is shown in [Wic, Fig. 11-13].

Minimum Distance

The **minimum distance** $d_{\min}$ of a rate-k/n convolutional code with constraint length K is d_K (that is, the CDF evaluated at $i = K$).

Example. The minimum distance for the code defined by the decoder in [Wic, Fig. 11-1] is $d_{\min} = d_4 = 3$.

The parameter $d_{\min}$ is useful for methods that use nK bits of a received word to decode a single bit.

Minimum Free Distance (1)

The **minimum free distance** d_{free} is the minimum Hamming distance between all pairs of complete convolutional codewords,

$$\begin{aligned} d_{\text{free}} &:= \min\{d(\mathbf{y}', \mathbf{y}'') \mid \mathbf{y}' \neq \mathbf{y}''\} \\ &= \min\{w(\mathbf{y}) \mid \mathbf{y} \neq \mathbf{0}\}. \end{aligned}$$

The minimum free distance is important in particular for the Viterbi decoder (to be considered), which uses the entire codeword to decode a single bit. It can be obtained by finding a cycle of minimum weight in the encoder graph starting and stopping at S_0.

Minimum Free Distance (2)

Example. The minimum free distance for the code defined by the decoder in [Wic, Fig. 11-1] is $d_{\text{free}} = 6$.

This and a previous example show the expected result that better performance is provided by techniques that use the entire word instead of nK bits to decode a single bit ($d_{\text{free}} = 6 > d_{\min} = 3$).

Some Results on Distance Parameters

- ▷ For noncatastrophic codes, $\lim_{i\to\infty} d_i = d_{\text{free}}$.
- ▷ Unlike linear block codes, *nonsystematic* convolutional codes often offer a higher minimum free distance than systematic codes of comparable constraint length and rate (cf. [Wic, Tables 11-1 and 11-2]).

Lists of nonsystematic rate-1/4, rate-1/3, rate-1/2, rate-2/3, and rate-3/4 convolutional codes with largest possible minimum free distance for small constraint lengths are listed in [Wic, Tables 11-3 to 11-7].

The Viterbi Decoding Algorithm

- In 1967, Viterbi proposed an algorithm for "asymptotically optimal" decoding of convolutional codes in memoryless noise.
- It turned out that the algorithm had been invented in operations research ten years earlier, when Minty presented an algorithm for a shortest-route problem.
- The Viterbi algorithm provides both a maximum-likelihood (ML) and a maximum a posteriori (MAP) decoding algorithm for convolutional codes.

Trellis Diagrams

A **trellis diagram** is an extension of the state diagram of a convolutional code, which explicitly shows the passage of time.

Example. A rate-1/3 encoder with two memory cells is shown in [Wic, Fig. 12-1] and its associated state diagram (which has $2^2 = 4$ states) in [Wic, Fig. 12-2]. In [Wic, Fig. 12-3], the state diagram is extended in time to form a trellis diagram. The edges of the trellis diagram are labeled with the corresponding output bits.

Some Properties of Trellis Diagrams (1)

- ▷ Every codeword in a convolutional code is associated with a unique path through the trellis diagram, starting and stopping at S_0.
- ▷ With total memory M and maximal memory order m, the trellis diagram has 2^M vertices at each time increment after time $t = m$.
- ▷ There are 2^k edges leaving each vertex. After time $t = m$, there are 2^k edges entering each vertex.
- ▷ Given an input sequence of kL bits, the trellis diagram has $L + m$ stages, and codewords have $n(L + m)$ bits.
- ▷ There are 2^{kL} different paths through a trellis diagram with $L + m$ stages.

Some Properties of Trellis Diagrams (2)

Example. The input sequence $\mathbf{x} = (011)$ to the encoder in [Wic, Fig. 12-1] is shown in [Wic, Fig. 12-4]. Since $L = 3$, $n = 3$, and $m = 2$, the codeword has $n(L + m) = 3(3 + 2) = 15$ bits and is $\mathbf{y} = (000, 111, 000, 001, 110)$.

The Viterbi Algorithm (1)

In a communication system, an information stream $\mathbf{x}$ is encoded into a convolutional codeword $\mathbf{y}$, which is transmitted across a (noisy) channel. At the receiving end, we want to find a good estimate $\mathbf{y}'$ of the transmitted word given the received word $\mathbf{r}$. Analogously to an earlier discussion for block codes,

- ▷ a maximum a posteriori (MAP) decoder selects an estimate $\mathbf{y}'$ that maximizes $p(\mathbf{y}' \mid \mathbf{r})$,
- ▷ a maximum likelihood (ML) decoder selects an estimate $\mathbf{y}'$ that maximizes $p(\mathbf{r} \mid \mathbf{y}')$, and
- ▷ these two decoders are identical when the distribution of the source words $\{\mathbf{x}\}$ is uniform.

The Viterbi Algorithm (2)

With a rate-k/n convolutional encoder and an input sequence $\mathbf{x}$ composed of L k-bit blocks

$$\mathbf{x} = (x_0^{(0)}, x_0^{(1)}, \ldots, x_0^{(k-1)}, x_1^{(0)}, x_1^{(1)}, \ldots, x_1^{(k-1)}, \ldots, x_{L-1}^{(k-1)}),$$

the output sequence $\mathbf{y}$ will consist of $L+m$ n-bit blocks (where m is the maximal memory order)

$$\mathbf{y} = (y_0^{(0)}, y_0^{(1)}, \ldots, y_0^{(n-1)}, y_1^{(0)}, y_1^{(1)}, \ldots, y_1^{(n-1)}, \ldots, y_{L+m-1}^{(n-1)}).$$

The corrupted word $\mathbf{r}$ that arrives at the receiver is

$$\mathbf{r} = (r_0^{(0)}, r_0^{(1)}, \ldots, r_0^{(n-1)}, r_1^{(0)}, r_1^{(1)}, \ldots, r_1^{(n-1)}, \ldots, r_{L+m-1}^{(n-1)})$$

The Viterbi Algorithm (3)

The decoder generates a maximum likelihood estimate

$$\mathbf{y}' = (y_0'^{(0)}, y_0'^{(1)}, \ldots, y_0'^{(n-1)}, y_1'^{(0)}, y_1'^{(1)}, \ldots, y_1'^{(n-1)}, \ldots, y_{L+m-1}'^{(n-1)}).$$

We assume that the channel is **memoryless**, that is, that the noise process affecting a given bit is independent of the noise process affecting any other bits. Then

$$p(\mathbf{r} \mid \mathbf{y}') = \prod_{i=0}^{L+m-1} \left(\prod_{j=0}^{n-1} p(r_i^{(j)} \mid y_i'^{(j)}) \right).$$

The Viterbi Algorithm (4)

For easier calculations, we take the logarithm of the previous expression (note that logarithms are monotonically increasing and $\log xy = \log x + \log y$) and transform it $x \to a(x+b)$ to get the **path metric**

$$M(\mathbf{r} \mid \mathbf{y}') = \sum_{i=0}^{L+m-1} \left(\sum_{j=0}^{n-1} M(r_i^{(j)} \mid y_i'^{(j)}) \right),$$

where

$$M(r_i^{(j)} \mid y_i'^{(j)}) = a[\log p(r_i^{(j)} \mid y_i'^{(j)}) + b].$$

The Viterbi Algorithm (5)

The sth **partial path metric** is defined as

$$M^s(\mathbf{r} \mid \mathbf{y}') = \sum_{i=0}^{s} \left(\sum_{j=0}^{n-1} M(r_i^{(j)} \mid y_i'^{(j)}) \right).$$

Until we reach the point in the trellis where more than one path enters each node, we assign to a node the value $M^s(\mathbf{r} \mid \mathbf{y}')$ of the only possible path. From that point on, the optimal partial path metric among the metrics for all of the entering paths is chosen. This is shown in [Wic, p. 296].

The Complete Viterbi Algorithm (1)

Let the vertex corresponding to state S_j at time t be denoted $S_{j,t}$. Each such vertex is assigned a value $V(S_{j,t})$ in the following way.

1. Set $V(S_{0,0}) = 0$ and $t = 1$.
2. At time t, compute the partial path metrics for all paths entering each vertex.
3. Set $V(S_{k,t})$ equal to the best partial path metric entering a vertex corresponding to state S_k at time t. One best edge survives (ties are randomly broken); the others are deleted from the trellis.
4. If $t < L + m$, $t := t + 1$ and goto Step 2; otherwise Stop.

The Complete Viterbi Algorithm (2)

Once all vertex values have been computed, the vector $\mathbf{y}'$ is obtained by starting at state S_0, time $t = L + m$ and following the surviving edges backward through the trellis.

Theorem 12-1. The path selected by the Viterbi decoder is the maximum likelihood path.

Hard-Decision Decoding

In **hard-decision decoding** each received bit is examined and a decision is made whether it represents a 0 or a 1. A **binary memoryless channel** model is depicted in [Wic, Fig. 12-6]. If $p(0 \mid 1) = p(1 \mid 0)$, we have a **binary symmetric channel (BSC)**.

The Viterbi Algorithm for the BSC (1)

For the BSC we have $p(0 \mid 1) = p(1 \mid 0) = p$ and $p(0 \mid 0) = p(1 \mid 1) = 1 - p$. By letting $a = (\log_2 p - \log_2(1-p))^{-1}$ (which is negative when $p < 1/2$) and $b = -\log_2(1-p)$,

$$M(r_i^{(j)} \mid y_i{}^{(j)}) = \frac{1}{\log_2 p - \log_2(1-p)}[\log_2 p(r_i^{(j)} \mid y_i{}^{(j)}) - \log_2(1-p)],$$

and we get the following table.

$M(r_i^{(j)} \mid y_i^{(j)})$	$r_i^{(j)} = 0$	$r_i^{(j)} = 1$
$y_i^{(j)} = 0$	0	1
$y_i^{(j)} = 1$	1	0

The Viterbi Algorithm for the BSC (2)

Since $a < 0$, we have a minimizing problem. For the BSC case, the path metric is simply the Hamming distance!

To get a maximizing problem, we can instead choose $a = (\log_2(1-p) - \log_2 p)$ and $b = -\log_2 p$; then we get the following metric.

$M(r_i^{(j)} \mid y_i^{(j)})$	$r_i^{(j)} = 0$	$r_i^{(j)} = 1$
$y_i^{(j)} = 0$	1	0
$y_i^{(j)} = 1$	0	1

Example: Viterbi Decoding for the BSC

When the encoder in [Wic, Fig. 12-1] is used, the sequence $\mathbf{x} = (110101)$ results in the codeword

$$\mathbf{y} = (111, 000, 001, 001, 111, 001, 111, 110).$$

Assume that when the word $\mathbf{y}$ is transmitted over a noisy channel, the following word is received:

$$\mathbf{r} = (1\overline{1}1, \overline{1}00, 001, 0\overline{1}1, 111, \overline{1}01, 111, 110).$$

In the expression of $\mathbf{r}$, a bar over a bit indicates an error. Using the second set of bit metrics for the BSC case, the result of the decoding is shown in [Wic, Fig. 12-8]. Several ties occur during decoding (for example, for S_3 at times $t = 3$ and $t = 5$); however, these have no impact on the path selected by the decoder.

Soft-Decision Decoding (1)

In **soft-decision decoding** the receiver takes advantage of side information in a received bit. Instead of assigning a 0 or a 1, as in hard-decision decoding, a more flexible approach is taken through the use of multibit quantization (cf. fuzzy logic!).

Four or more decision regions are established, ranging from a *strong one* to a *strong zero*.

Soft-Decision Decoding (2)

Let the transmitted bits $\{y_i^{(j)}\}$ take the values ± 1. The path metric for the AWGN channel are then the inner product of the received word and the codeword. The individual bit metrics are

$$M(r_i^{(j)} \mid y_i^{(j)}) = r_i^{(j)} y_i^{(j)}.$$

Maximization of the path metric is equivalent to finding the codeword $\mathbf{y}$ that is closest to $\mathbf{r}$ in terms of Euclidean distance (note: for the BSC, Hamming distance was considered).

In dealing with Euclidean distance and real numbers, finite precision of digital computers will have an impact on the result. In practice, however, this impact turns out to be negligible.

Discrete Symmetric Channels

The fact that the received signal can take more than two values is modeled as a **discrete symmetric channel**. Such a channel with four values for the received signal is depicted in [Wic, Fig. 12-10]. The four values are

$\underline{0}$ strong 0,
0 weak 0,
1 weak 1, and
$\underline{1}$ strong 1.

Example: Soft-Decision Viterbi Decoding (1)

Decoding is performed almost exactly as in the hard-decision case, the only difference being the increased number (and resolution) of the bit metrics.

We consider the following conditional probabilities in [Wic, Fig. 12-10].

$p(r \mid y)$	$r = \underline{0}$	$r = 0$	$r = 1$	$r = \underline{1}$
$y = 0$	0.50	0.32	0.13	0.05
$y = 1$	0.05	0.13	0.32	0.50

Example: Soft-Decision Viterbi Decoding (2)

$\log_2 p(r \mid y)$	$r = \underline{0}$	$r = 0$	$r = 1$	$r = \underline{1}$
$y = 0$	-1.00	-1.64	-2.94	-4.32
$y = 1$	-4.32	-2.94	-1.64	-1.00

With $M(r \mid y) = 1.5[\log_2 p(r \mid y) - \log_2(0.05)]$, we get the following table.

$M(r \mid y)$	$r = \underline{0}$	$r = 0$	$r = 1$	$r = \underline{1}$
$y = 0$	5	4	2	0
$y = 1$	0	2	4	5

Example: Soft-Decision Viterbi Decoding (3)

As in a previous example, assume that the word

$$\mathbf{y} = (111, 000, 001, 001, 111, 001, 111, 110)$$

is transmitted, and that the received word is

$$\mathbf{r} = (1\underline{\overline{0}}1, \overline{\underline{1}}\underline{0}\overline{0}, 0\underline{0}\underline{1}, \underline{0}\overline{1}1, \underline{1}\underline{1}\overline{0}, \overline{1}\overline{1}\overline{0}, 1\underline{1}\underline{1}, 1\underline{1}0).$$

The soft-decoding trellis diagram and the result of the decoding are shown in [Wic, Fig. 12-11].