"Back to coding"

Re-visit: Block codes
New: Convolutional codes / Viterbi algorithm

1. \((n,k,d)\)-code: A set of \(2^k\) codewords of length \(n\)

\[
\text{message} \xrightarrow{\text{coder}} \text{codeword} \quad (+\text{the coder function})
\]

\(d = \min \text{ HD between codewords}\)

* An \((n,k,d)\)-code:
  - corrects \((d-1)\%2\) bit errors
  - detects up to \(d-1\) bit errors

Ex. 1
How many errors will a \((15, 11, 3)\) code detect?
Answer: 2

Ex. 2
\((20, 8, 4)\) code correct?
Answer: 1

2. \((? , k, 3)\): How many extra bits needed?

Th. \(m\) extra bits enough only when \(k \leq 2^m - m - 1\)

With good coding, this is achievable:
"Hamming code": \((2^m - 1 , 2^m - m - 1, 3)\)

(Proof of Th.: \((k+m+1)2^k \leq 2^{k+m}\))

Ex. 3
For a 100-bit message, how many extra parity bits needed to get single error correction?

\(2^6 - 6 - 1 = 63 < 100\)
\(2^7 - 7 - 1 = 120 > 100\) \(\Rightarrow\) Answer: 7
3. **Linear Code:** When codeword + code word = another codeword

**Ex. 4:** is \( \begin{bmatrix} 0011 \\ 0110 \\ 0101 \\ 1001 \end{bmatrix} \) linear? \( \begin{bmatrix} 000 \end{bmatrix} \) ?

**Ex. 5:** Is it possible for a \( \mathbb{F}_2 \)-linear code to have exactly 17 words?

**Convolutional Codes:** (As if) using one bit taking codeword for each data sequence.

\( b[n] \) \( \xrightarrow{\text{data}} \) \( \xrightarrow{\text{bits}} \) \( \xrightarrow{\text{Encoded}} \) \( \xrightarrow{\text{\( \Gamma \)-words}} \) \( \xrightarrow{\text{sequence}} \) \( w[n] \)

**Specifically Convolutional Codes**:

\( b[n] \) \( \xrightarrow{\text{(Single} \ \mathbb{F}_2 \ \text{Input)}} \) \( \xrightarrow{\mathbb{F}_2 \text{-LTI}} \) \( \xrightarrow{\mathbb{F}_2 \text{-LTI}} \) \( \xrightarrow{\mathbb{F}_2 \text{-LTI}} \) \( \ldots \)

\( \xrightarrow{\text{Multiple} \ \mathbb{F}_2 \ \text{Outputs}} \)

**Ex. 6:** \( \Gamma = 3 \)

\( h_1 = \begin{bmatrix} 1 & 1 & 1 & 1 \end{bmatrix} \)

\( h_2 = \begin{bmatrix} 1 & 1 & 1 & 1 \end{bmatrix} \)

\( h_3 = \begin{bmatrix} 1 & 1 & 1 & 1 \end{bmatrix} \)

\( b[0] \) \( b[1] \) \( b[2] \)

\( \begin{bmatrix} 0, 0, 0, 1, 1, 0, 1, 0, \ldots \end{bmatrix} \) \( \xrightarrow{\text{Enc.}} \) \( \begin{bmatrix} 1, 0, 1, 1, 1, 0, \ldots \end{bmatrix} \)

\( \begin{bmatrix} 1, 1, 1, 0, 0, \ldots \end{bmatrix} \)

\( \begin{bmatrix} 0, 1, 0, 1, 1, \ldots \end{bmatrix} \)

\( \begin{bmatrix} 110, 011, 110, 101, 001, \ldots \end{bmatrix} \)
To decode: need "best approximation" deconvolution:

\[ X \rightarrow \text{LTI} \rightarrow Y \]

Given \( h \) and \( Y_0 \), find \( X \) such that \( h * X \) is the best approximation of \( Y_0 \).

We could have done this for the original convolution (with \( \mathbb{R} \) instead of \( \mathbb{F}_2 \)), but the quantized case is easier.

Main problem: a direct search through all possible sequences "X" is too long.

Ex. 7 If \( Y \) is 100 words long, how many possible "X"?

Answer: \( 2^{100} \approx 10^{30} \)

Can we divide the search into smaller pieces?

Start here \( \rightarrow Y_0 \)

\( \text{If we knew } X_A \text{ such that } h * X_A \right|_{0-49} \text{ is the best approximation of } Y_A, \) would it help?

\( \uparrow \)

Solution:
Figure out what is exactly the "current memory", and optimize \( X_A \) while fixing the "memory".

Ex. 8 What is the memory in Ex. 6?

Answer: the 4 previous bits:

\[
\text{MEMORY}_0 = \text{STATE}[n] = \begin{bmatrix}
b[n-1] \\
b[n-2] \\
b[n-3] \\
b[n-4]
\end{bmatrix}
\]
"STEP K" TASK:

For every possible value $x$ of state $[k]$, find the minimal distance between $y_0|_{0-k}$ and $y_1|_{0-k}$, where $y_1|_{0-k}$ is the encoder's response to an input which brings the encoder's state to $x$ in $k$ steps.

**Dynamic Programming:**

Once the "STEP K" task is solved, a solution of the "STEP K+1" task is easy. Solve "STEP 0" task first. Then: solve "STEP 1" task. Then: solve "STEP 2" task ...

Once all steps are done, recovering the optimal input is easy.

**Ex. 9**

$x[0] \xrightarrow{\text{Enc.}} y_1[n] = x[n]$

$y_2[n] = x[n] + x[n-1]$ (in $\mathbb{F}_2$)

($\Gamma = 2$, $K = 1$, $h_1[n] = \delta[n]$, $h_2[n] = \delta[n] + \delta[n-1]$)

"STATE" $[n] = x[n-1]$


$y[0]$, $y[1]$, $y[2]$, ...
Ex. 9 (cont.)

Assume 01 00 10 11 01 : RECEIVED

Decoded so far: 0,0,0,1,0

\[ \rightarrow 00 00 00 11 01 \]