Spreading Codes and Characteristics

and

Error Correction Codes

Global Navigational Satellite Systems (GNSS-16)

Short course, NERTU

Prasad Krishnan

International Institute of Information Technology, Hyderabad

December, 2016

Spreading Codes

Talk Outline - Spreading Codes

- Spreading codes idea
- Need for spreading codes in GNSS
- Generating spreading codes for GNSS

What are Spreading Codes?

Example

- Consider that you have a message bit b (can be zero or one).
- Instead of transmitting b we transmit - $b[1\ 0\ 0\ 0\ 0]$.
- The vector [1 0 0 0 0] is like a carrier - we call it the *code*.
- To decode, multiply received vector by $[1\ 0\ 0\ 0\ 0]^T$.

What are Spreading Codes?

Example

- 4 different bits with 4 **orthogonal** codes transmitted at the same time.
- $$\begin{aligned} & b_0[1\ 0\ 0\ 0] + b_1[0\ 1\ 0\ 0] + b_2[0\ 0\ 1\ 0] + b_3[0\ 0\ 0\ 1] \\ = & b_0\boldsymbol{x_0} + b_1\boldsymbol{x_1} + b_2\boldsymbol{x_2} + b_3\boldsymbol{x_3} \end{aligned}$$

What are Spreading Codes?

Example

- 4 different bits with 4 **orthogonal** codes transmitted at the same time.
-

$$
\begin{aligned}
& b_0[1\ 0\ 0\ 0] + b_1[0\ 1\ 0\ 0] + b_2[0\ 0\ 1\ 0] + b_3[0\ 0\ 0\ 1] \\
= & b_0\boldsymbol{x_0} + b_1\boldsymbol{x_1} + b_2\boldsymbol{x_2} + b_3\boldsymbol{x_3}
\end{aligned}
$$

- Note that $\boldsymbol{x_i}\boldsymbol{x_j}^T = \begin{cases} 1 & \text{if } i = j \\ 0 & \text{if } i \neq j \end{cases}$
- Can get the bit b_i by multiplying with $\boldsymbol{x_i}^T$. (i represents the shift in time).

What are Spreading Codes? - Finding Delays

Example

- Consider that a coded bit b comes with an arbitrary unknown delay $(0 \leq j \leq 3)$.
- Received vector $= b\boldsymbol{x_j}$ $(0 \leq j \leq 3)$.
- Then can we find out the delay j and bit b?
- Multiplying with $\boldsymbol{x_i}$ $(0 \leq i \leq 3)$ does the trick.

More generally - Pseudo-Random-Noise Sequences

- Consider two or more binary sequences ($\{+1, -1\}$) sequence which have 'good' autocorrelation properties and 'good' cross correlation properties.
- Such sequences are called Pseudo-Random-Noise sequences.

More generally - Pseudo-Random-Noise Sequences

- Consider two or more binary sequences ($\{+1, -1\}$) sequence which have 'good' autocorrelation properties and 'good' cross correlation properties.
- Such sequences are called Pseudo-Random-Noise sequences.

'Good' autocorrelation property of a sequence $\boldsymbol{x}$

- $\boldsymbol{x}_i \boldsymbol{x}_j^T$ has a high value if $i = j$ and low value if $i \neq j$.

'Good' cross-correlation property of sequences $\boldsymbol{x}$ and $\boldsymbol{y}$

- $\boldsymbol{x}_i \boldsymbol{y}_j^T$ has a low value for any i, j.

How are PNR sequences useful in GNSS?

Each satellite has its own unique PRN sequence, and uses it to modulate data transmitted to receivers.

Ranging

- Good autocorrelation properties $\rightarrow$ Find Delay due to separation between Rx and Satellite Tx.
- Delay (from multiple satellites) $\rightarrow$ User Location.

How are PNR sequences useful in GNSS?

Each satellite has its own unique PRN sequence, and uses it to modulate data transmitted to receivers.

Ranging

- Good autocorrelation properties $\rightarrow$ Find Delay due to separation between Rx and Satellite Tx.
- Delay (from multiple satellites) $\rightarrow$ User Location.

Saving spectrum

- Good cross-correlation properties $\rightarrow$ Decode info from different satellites.
- Multiple satellites can transmit over the same frequency.

How are PNR sequences useful in GNSS?

Each satellite has its own unique PRN sequence, and uses it to modulate data transmitted to receivers.

Ranging

- Good autocorrelation properties → Find Delay due to separation between Rx and Satellite Tx.
- Delay (from multiple satellites) → User Location.

Saving spectrum

- Good cross-correlation properties → Decode info from different satellites.
- Multiple satellites can transmit over the same frequency.

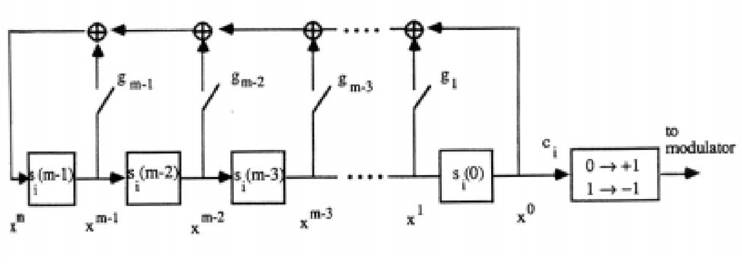
Gain in SNR

- Same bit is encoded in a long PRN sequence → redundancy.
- Redundancy → provides SNR gain (and hence lowers prob. of error).

Generating 'Pseudo-Random-Noise'

- PN Sequences deterministically generated, yet possess properties of randomly generated sequences
- PN sequences generated using *linear feedback shift registers*.

Linear Feedback Shift Registers



- Output sequence is given by

$$c_{i+m} = g_{m-1}c_{i+m-1} + g_{m-2}c_{i+m-2} + \ldots + g_1 c_{i+1} + c_i (mod\ 2)$$

Characteristic Polynomial of LFSR

- Since the operation is binary addition, the above output equation can be rewritten as

$$\sum_{\ell=0}^{m} g_\ell c_{i+\ell} = 0,$$

where $g_0 = g_m = 1$

- The characteristic polynomial of the LFSR is given by

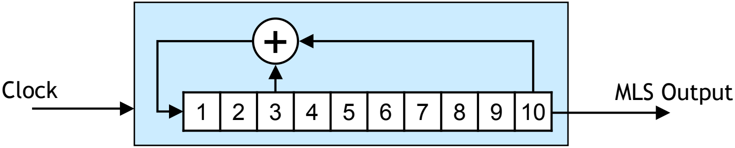
$$g(x) = \sum_{\ell=0}^{m} g_\ell x^\ell$$

- We are interested in special kinds of LFSRs which can output PRN sequences.

Primitive Polynomial

- Every polynomial $g(x)$ with coefficients in binary field having $g(0) = 1$ divides $x^N + 1$ for some N. The smallest N for which this is true is called the period of $g(x)$.
- An irreducible polynomial of degree m whose period is $2^m - 1$ is called a primitive polynomial

m-Sequences : Examples of PRN sequences



- An LFSR produces an *m*-sequence (maximum length) if and only if its characteristic polynomial is a primitive polynomial
- In the above example, the polynomial is $g(x) = 1 + x^3 + x^{10}$

Delay and Add Property of m-Sequences

- The cyclic shift of an *m*-sequence is also an *m*-sequence
- The sum of an *m*-sequence and a cyclic shift of itself is also an *m*-sequence

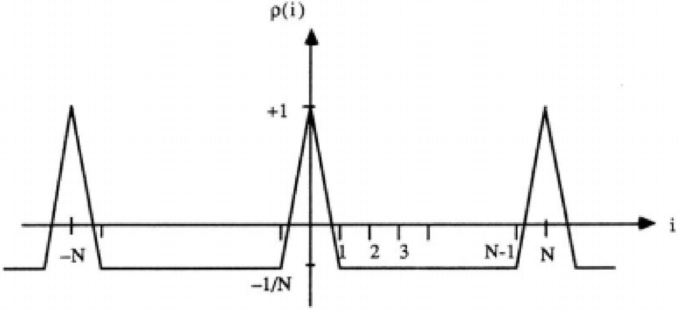
Autocorrelation Function of m-Sequences

Let (s_t) be an m-sequence of period $N = 2^n - 1$ Then the autocorrelation of the m-sequence is

$$\theta_{s,s}(\tau) = \begin{cases} 2^n - 1 & if \ \tau = 0(mod 2^n - 1) \\ -1 & if \ \tau \neq 0(mod 2^n - 1) \end{cases}$$

Plot of Autocorrelation Function

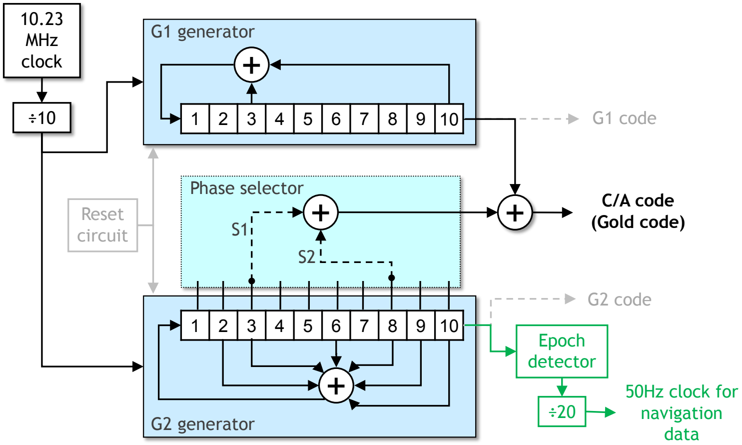
Normalized autocorrelation function



Gold Sequences

- m-Sequences have good auto-correlation properties but poor cross-correlation properties (cross-correlation can be high, which we don't want).
- Two m-Sequence generators are used to generate a "Gold Sequence" which has good cross-correlation properties

Generating Gold Sequences



Gold Sequences in GPS

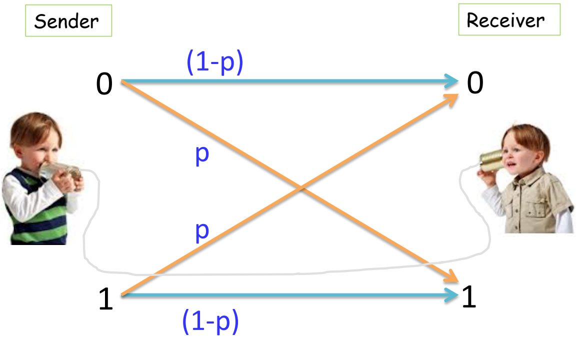
Satellite ID Number Number	GPS PRN Signal Number	Code Phase Selection (G2)	Code Delay Chips	First 10 Chips Octal
1	1	$2\oplus 6$	5	1440
2	2	$3\oplus 7$	6	1620
3	3	$4\oplus 8$	7	1710
4	4	$5\oplus 9$	8	1744
5	5	$1\oplus 9$	17	1133
6	6	$2\oplus 10$	18	1455
7	7	$1\oplus 8$	139	1131
8	8	$2\oplus 9$	140	1454
9	9	$3\oplus 10$	141	1626
10	10	$2\oplus 3$	251	1504
11	11	$3\oplus 4$	252	1642
12	12	$5\oplus 6$	254	1750
13	13	$6\oplus 17$	255	1764
14	14	$7\oplus 8$	256	1772
15	15	$8\oplus 9$	257	1775
16	16	$9\oplus 10$	258	1776
17	17	$1\oplus 4$	469	1156
18	18	$2\oplus 5$	470	1467
19	19	$3\oplus 6$	471	1633
20	20	$4\oplus 7$	472	1715
21	21	$5\oplus 8$	473	1746
22	22	$6\oplus 9$	474	1763
23	23	$1\oplus 3$	509	1063
24	24	$4\oplus 6$	512	1706
25	25	$5\oplus 7$	513	1743
26	26	$6\oplus 8$	514	1761
27	27	$7\oplus 9$	515	1770
28	28	$8\oplus 10$	516	1774
29	29	$1\oplus 6$	859	1127
30	30	$2\oplus 7$	860	1453
31	31	$3\oplus 8$	861	1625
32	32	$4\oplus 9$	862	1712
-	33	$5\oplus 10$	863	1745
-	34	$4\oplus 10$	950	1713
-	35	$1\oplus 7$	947	1134
-	36	$2\oplus 8$	948	1456
-	37	$4\oplus 10$	950	1713

Forward Error Correction

Talk Outline - Error Correcting Codes

- Need for Error Correction
- Error Correcting Codes (general principles and examples)
- Encoding and Decoding of a Block Code (Hamming Code)
- Types of Error Correcting Codes (in GNSS)
- Encoding and Decoding of Convolutional Codes

Channel Coding



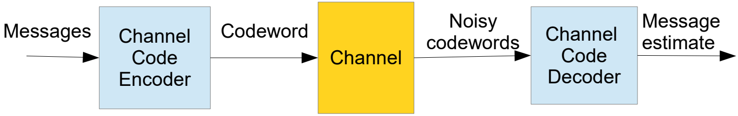
p : cross-over probability, say 0.1

- The bit cross-over probability p (< 0.5) is a property of the channel.
- Free to manipulate the input and output to the channel.
- Encode messages to *codewords* (add redundancy cleverly) : Reduce effective $Prob(error)$.

A trivial code example - Repetition code

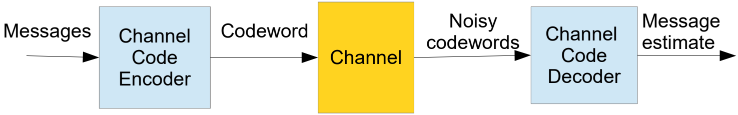
- Repeat the same bit three times
- Message 0 $\rightarrow$ [0 0 0] (codeword), Message 1 $\rightarrow$ [1 1 1].
- Decode by Majority logic.
- For the above channel, probability of error comes down (Check!).

General ideas behind FEC for the binary symmetric channel



- Decoder decides the Tx codeword $\boldsymbol{c}$ based on received Noisy Codeword $\boldsymbol{y}$.
- Decoding rule : Decoding for the **most likely codeword.** (Choose that $\boldsymbol{c}$ which maximizes $p(\boldsymbol{y}|\boldsymbol{c})$).

General ideas behind FEC for the binary symmetric channel



- Decoder decides the Tx codeword $\boldsymbol{c}$ based on received Noisy Codeword $\boldsymbol{y}$.
- Decoding rule : Decoding for the **most likely codeword.** (Choose that $\boldsymbol{c}$ which maximizes $p(\boldsymbol{y}|\boldsymbol{c})$).
- Probability that every transmitted bit is flipped is $p < 0.5$
- If you don't code at all, the rule decodes to the bit that was received as it is.

General ideas behind FEC for the binary symmetric channel

- For a code of length n: Choose the codeword which is closest to Rx Vector $\boldsymbol{y}$ in terms of number of flipped bits.
- Minimum Hamming Distance Rule.

General ideas behind FEC for the binary symmetric channel

- For a code of length n: Choose the codeword which is closest to Rx Vector $\boldsymbol{y}$ in terms of number of flipped bits.
- Minimum Hamming Distance Rule.

Example

Repetition Code

- The Majority Decoder is infact the Minimum Hamming Distance Decoder.
- The Repetition Code can correct any single-bit error.

General ideas behind FEC for the binary symmetric channel

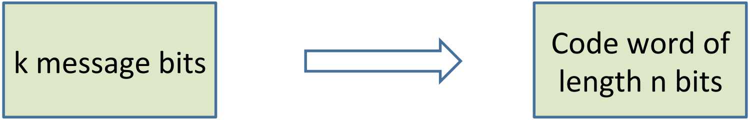
- For a code of length n: Choose the codeword which is closest to Rx Vector $\boldsymbol{y}$ in terms of number of flipped bits.
- Minimum Hamming Distance Rule.

Example

Repetition Code

- The Majority Decoder is infact the Minimum Hamming Distance Decoder.
- The Repetition Code can correct any single-bit error.
- Increase n to decrease P(error).
- Rate $\frac{1}{n}$: For 1 bit of message, we need to send n coded bits. (Pretty bad!)

Can we do better? - Linear Block Codes



- Each set of k message bits maps to a unique codeword
- Each of the n bits is a linear combination of k message bits
- n = length of the code, k = dimension of the code

Linear Block Codes

- Let $\boldsymbol{u}$ be the message vector and $\boldsymbol{c}$ is the corresponding codeword.
- How to get $\boldsymbol{c}$ from $\boldsymbol{u}$?
- Linear Block Codes used a Linear Map

$$\boldsymbol{c} = \boldsymbol{u}G$$

- G is a full-rank matrix of size $k \times n$ $(k \leq n)$. The code is a (n, k) *linear block code*.

Linear Block Codes

- Let $\boldsymbol{u}$ be the message vector and $\boldsymbol{c}$ is the corresponding codeword.
- How to get $\boldsymbol{c}$ from $\boldsymbol{u}$?
- Linear Block Codes used a Linear Map

$$\boldsymbol{c} = \boldsymbol{u}G$$

- G is a full-rank matrix of size $k \times n$ $(k \leq n)$. The code is a (n, k) *linear block code*.

Repetition Code

- $G = (1\ 1\ 1)$.
- Message $\boldsymbol{u} \in \{1, 0\}$.
- Codewords are [1 1 1] and [0 0 0].

Linear Block Codes - Examples

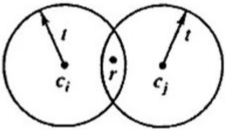
- Hamming Codes - a class of single error correcting codes.
- Reed Solomon Codes.

Example

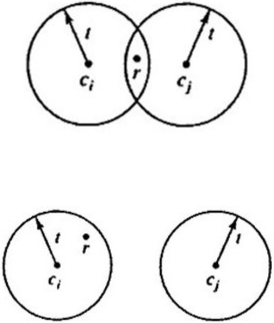
The $(7,4)$ Hamming Code

- $$G = \begin{pmatrix} 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 1 \end{pmatrix}.$$

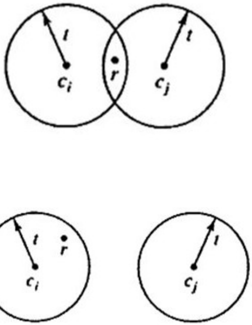
Error Correcting Capability of a Block Code



Error Correcting Capability of a Block Code



Error Correcting Capability of a Block Code



- A block code can correct t errors if and only if 'Hamming' balls of size t around codewords don't intersect.
- Minimum distance of the code must be at least $2t + 1$.
- Linearity $\implies$ Minimum weight of the code is at least $2t + 1$.

Decoding - The Parity Check Matrix

- The Parity Check Matrix : A full-rank $n - k \times n$ matrix such that $GH^T = 0$.

- For the Hamming Code :

$$H = \begin{pmatrix} 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 1 \end{pmatrix}.$$

Decoding

- Received vector $\boldsymbol{y} = \boldsymbol{c} + \boldsymbol{e}$.
- Compute $\boldsymbol{s} = \boldsymbol{y}H^T = \boldsymbol{c}H^T + \boldsymbol{e}H^T = \boldsymbol{x}GH^T + \boldsymbol{e}H^T = \boldsymbol{e}H^T$.
- Corresponding to any error vector of weight upto t there is an unique syndrome.

Syndrome Decoding

Syndrome decoding for errors of weight upto t.

1. Find the syndrome $\boldsymbol{s}$
2. Find $\boldsymbol{e}$ corresponding to $\boldsymbol{s}$.
3. Find $\boldsymbol{c} = \boldsymbol{r} - \boldsymbol{e}$. Map it back to $\boldsymbol{x}$.

Types of Codes used in GNSS

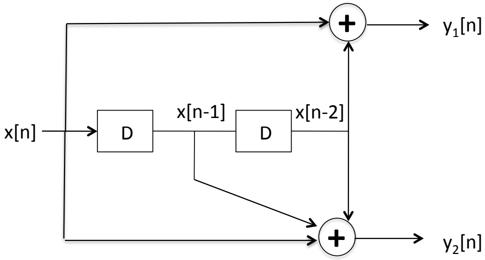
Table 2. Channel coding comparison

Coding	NAV	CNAV	Galileo	CNAV-2
Hamming	Yes	No	No	No
Convolution	No	Yes	Yes	No
CRC	No	Yes	Yes	Yes
Interleaving	No	No	Yes	Yes
LDPC	No	No	No	Yes
BCH	No	No	No	Yes

Convolutional Encoding

- Convolutional codes are used in applications that require good performance with low encoding complexity.
- Convolution codes have memory that utilises previous bits to encode or decode following bits (block codes are memoryless)

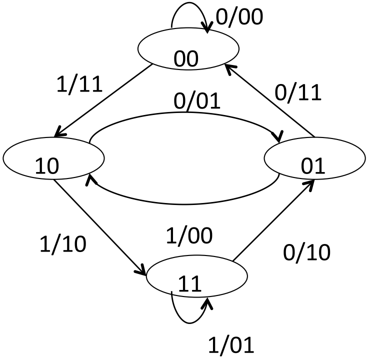
Convolutional Encoding



Encoder

- $y_1[n] = x[n] \oplus x[n-1] \oplus x[n-2]$
- $y_2[n] = x[n] \oplus x[n-2]$
- Rate $\frac{1}{2}$ convolutional encoder
- Constraint length for each input is 2

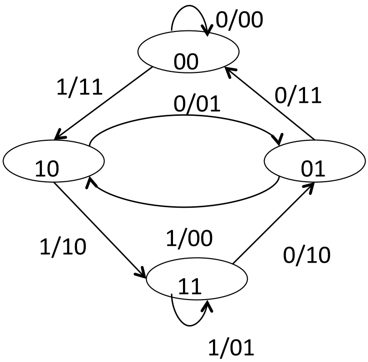
State Diagram



State diagram

- State transitions are given by input/output

Example Encoding



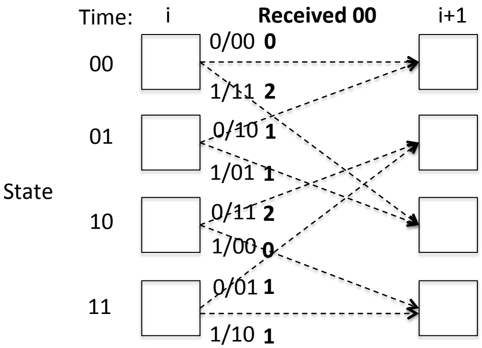
State diagram

- Input: 010111001010001
- Output: 00 11 10 00 01 10 01 11 11 10 00 10 11 00 11

Brute Force Approach

- Going through the list of possible transmit sequences and comparing Hamming distance is highly complex
- A transmit sequence of N bits has 2^N possible strings, exponential complexity
- Low Complexity Decoder: Viterbi Decoder - decoding on trellis

Branch Metric



- The branch metric for hard decision decoding. In this example, the receiver gets the parity bits 00
- Two of the branch metrics are 0, corresponding to the only states and transitions where the corresponding Hamming distance is 0
- Other non-zero branch metrics correspond to cases where there are bit errors

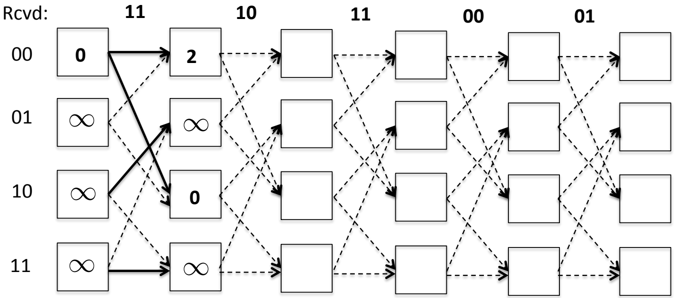
Computing Path Metric

- Value of $\text{PM}[s, i]$ - total number of bit errors detected when comparing the received parity bits to the most likely transmitted message, considering all messages that could have been sent by the transmitter until time step i

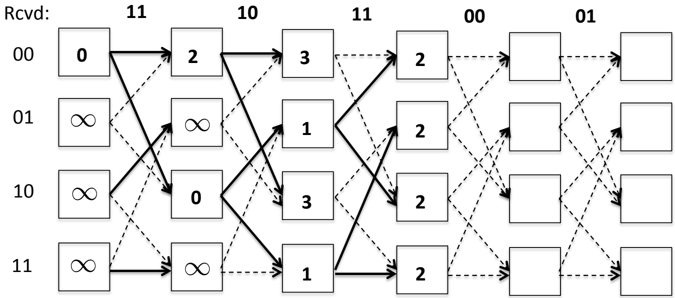
- If the transmitter is at state s at time step $i + 1$, then it must have been in only one of two possible states at time step i, say α and β
- Path Metric update is given by

$$\text{PM}[s, i+1] = \min(\text{PM}[\alpha, i] + \text{BM}[\alpha \rightarrow s], \text{PM}[\beta, i] + \text{BM}[\beta \rightarrow s])$$

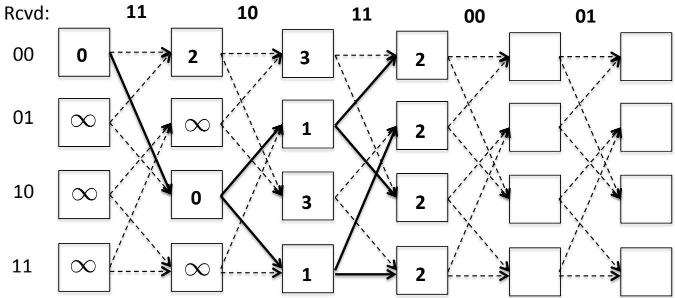
Viterbi Decoding: Step 1



Viterbi Decoding: Step 2

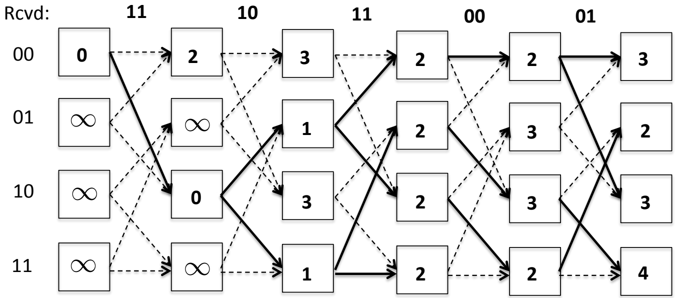


Viterbi Decoding: Step 3

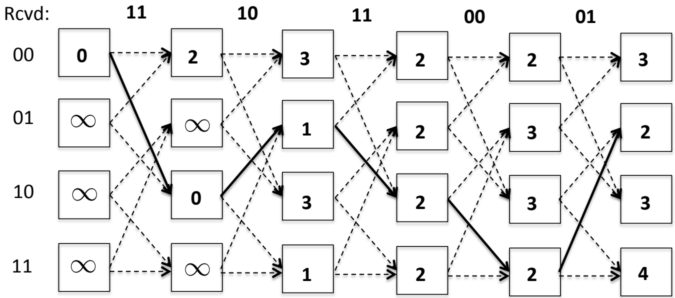


- Showing only survivor paths

Viterbi Decoding: Step 4



Viterbi Decoding: Step 5



- To produce the message, start from final state with smallest path metric and word backwards and then reverse the bits

Hard Decision Decoding

- Hard decision decoding digitizes the received voltagee signals by comparing it to a threshold, before passing it to the decoder
- Loss of Information
- 0.500001 and 0.99999 are both treated as “1” by the decoder even it is more likely that 0.99999 is a “1”
- Hamming distance as branch metric

Soft Decision Decoding

- Soft Decision Decoding does not digitise the incoming samples prior to decoding
- If the convolutional code produces p parity bits and p corresponding analog samples are $v = v_1, v_2, \ldots, v_p$, a soft decision branch metric is given by

$$BM_{\text{soft}}[u, v] = \sum_{i=1}^{p} (u_i - v_i)^2$$

Thanks!