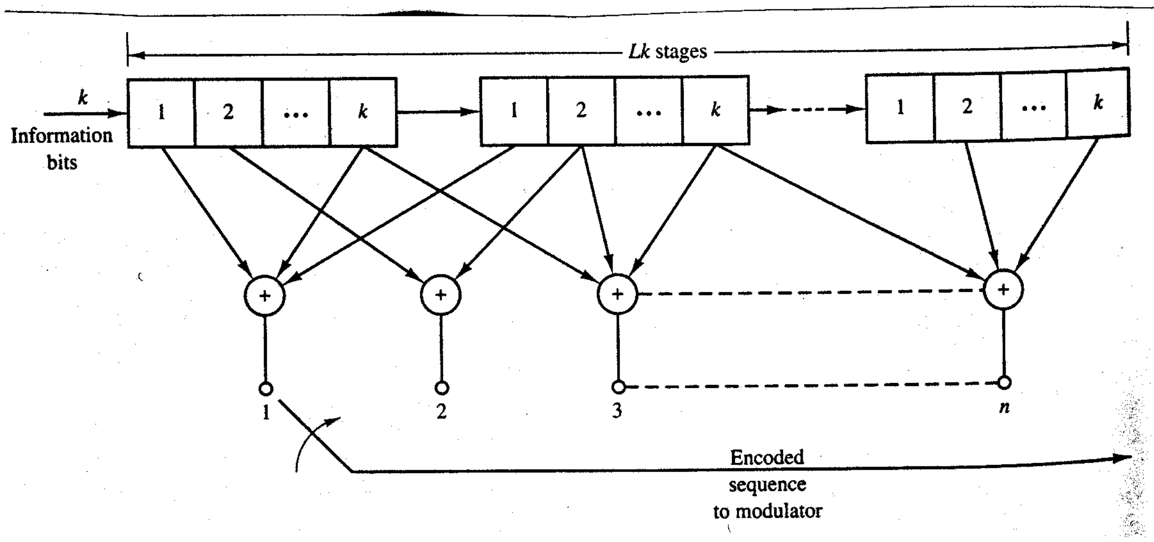
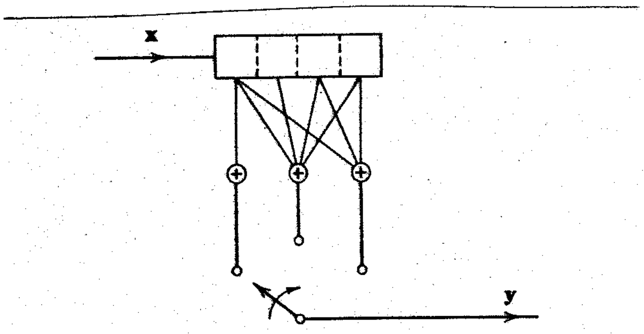
8.7 Convolutional Codes

- Unlike block codes, convolutional codes have no fixed length, although they are still linear.

- Generated by a shift register of Lk stages: read in k bits, read out n bits, constraint length L



specify connections by generator sequences g_1 to g_n; each sequence Lk bits long to identify if connected.



$k=1$, $n=3$, $r=\frac{k}{n}=\frac{1}{3}$

$L=4$

$g_1 = 1000 = 10_8$

$g_2 = 1111 = 17_8$

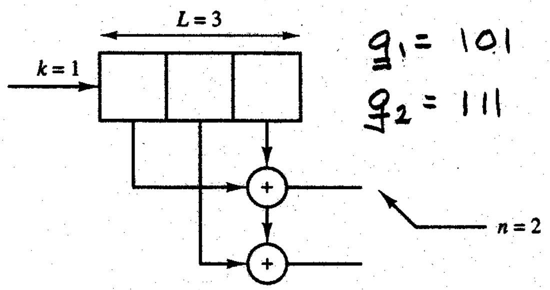
$g_3 = 1011 = 13_8$

- Consequences
  - No limit on transmitted info length – flexible
  - Last k info bits clocked in still have to affect Ln output bits for reasonable error protection. So attach a tail of $k(L-1)$ zero bits to flush out the encoder. For I info bits, send

$$N = \frac{I}{k} \cdot n + (L-1)n \text{ bits, so}$$

$$\text{rate} = \frac{I}{N} = \frac{I}{\frac{I}{k} \cdot n + (L-1)n} \approx \frac{k}{n}$$

  - Each output bit position is a convolution of info bits (decimated) with its own unit pulse response.



$g_1 = 101 = x^2 + 1$

$g_2 = 111 = x^2 + x + 1$

  - It's linear: if $\underline{i}_1 \rightarrow \underline{c}_1$ and $\underline{i}_2 \rightarrow \underline{c}_2$, then

$$\underline{i}_1 + \underline{i}_2 \longrightarrow \underline{c}_1 + \underline{c}_2$$

Also $\underline{c}_i + \underline{c}_j = \underline{c}_k$ all codewords

- Convolutional codes can be represented as a tree

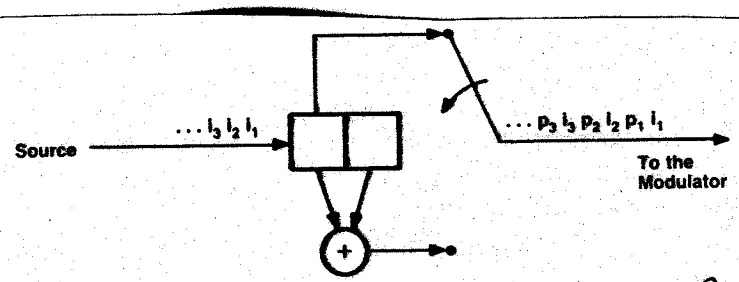


FIGURE 8.1. Encoder for a simple convolutional code. $L = 2$

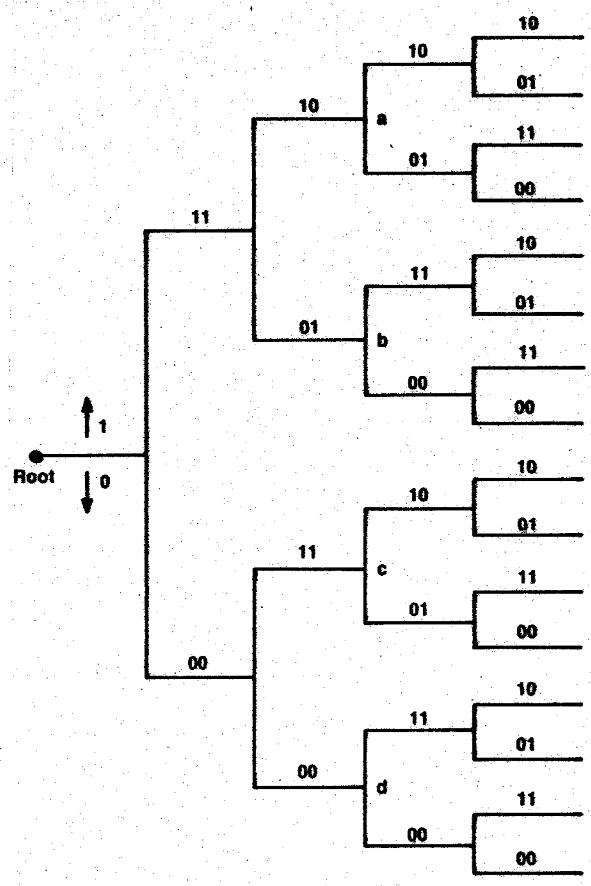


FIGURE 8.2. A simple code tree.

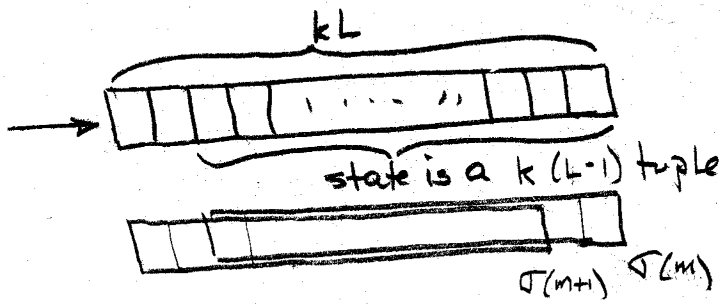
$$\left.\begin{array}{l} \underline{g}_1 = 10 \\ \underline{g}_2 = 11 \end{array}\right\} \text{a rate } \tfrac{1}{2} \text{ code}$$

- Tree codes are more general than convolutional codes — not nec. linear

- Alternative view: A finite state machine with states σ_ℓ, $\ell = 0 \ldots 2^{k(L-1)}$ in which

$$\sigma(m) = f(\underline{i}(m), \sigma(m-1))$$

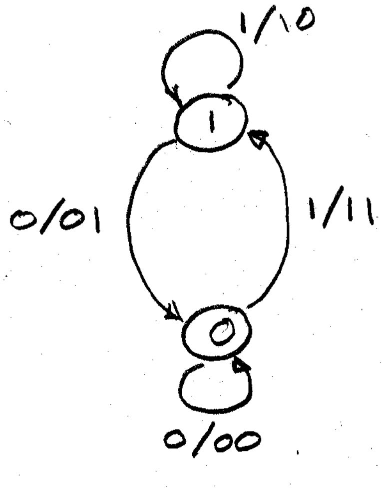
$$\underline{c}(m) = g(\sigma(m-1), \sigma(m))$$



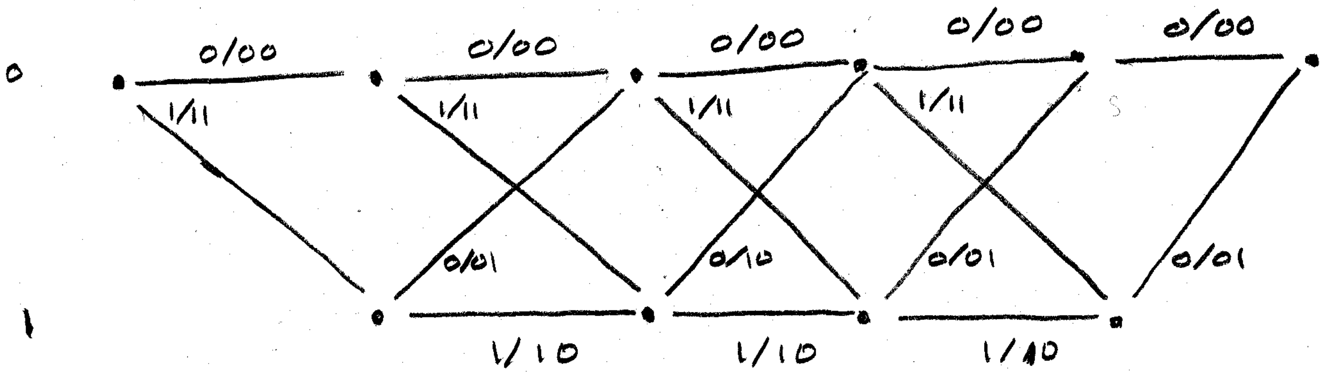
shown for k = 2

shown for new bits entering from left

For the simple encoder of the previous page ($n=2$, $k=1$, $L=2$) the state is defined by the rightmost 1 bit. State diagram:

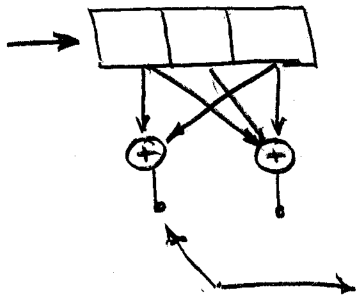


- Trellis diagram unrolls the state diagram

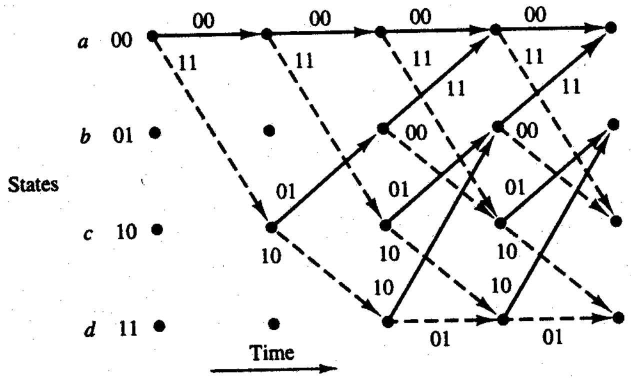


- begins and ends in known states, with starting "transient" and flushing out shift reg at the end
- as before, determining the state sequence
  $\sigma(0), \sigma(1), \sigma(2), \ldots$
  is equivalent to decoding the info bits

- A more elaborate code $n=2$, $k=1$, $L=3$



What defines the state?



Decoding Convolutional Codes

- Suppose we make hard decisions, resulting in the binary sequence $\underline{y}$. We'd like to choose the code word closest to $\underline{y}$.

Justification for this criterion on ML grounds.

- With the Hamming distance as the metric to minimize, we notice that the overall Hamming distance is the sum of non-negative Hamming distances on each transition. Hence branch metric at time m is

$$\mu(\underline{y}(m), \sigma(m-1), \sigma(m)) = d_H(\underline{y}(m), \underline{g}(\sigma(m-1), \sigma(m)))$$
$$= w_H(\underline{y}(m) + \underline{g}(\sigma(m-1), \sigma(m)))$$

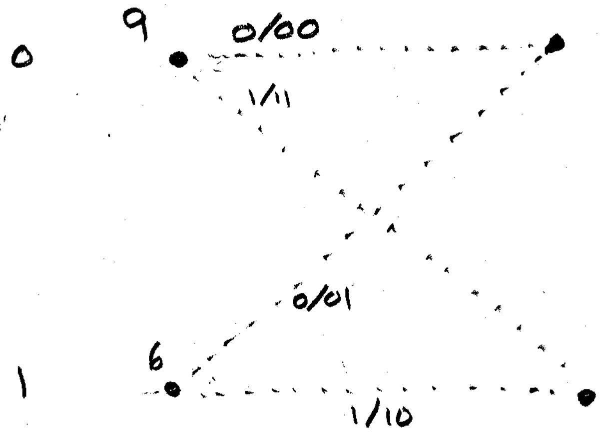
- This is custom-made for the Viterbi algorithm.

Example: We have the metrics from time $m-1$

$M(\sigma(m-1)=0) = 9$ and receive $\underline{y}(m) = 01$

$M(\sigma(m-1)=1) = 6$

Update survivors and cumulative metric



- We can operate with soft decisions almost as easily. The metric for the codeword in total is, as usual, the Euclidean distance, or squared Euc. distance. Let the received signal be $\underline{r}$, transmitted be $\underline{s}(\underline{c})$ (depends on modulation). Choose

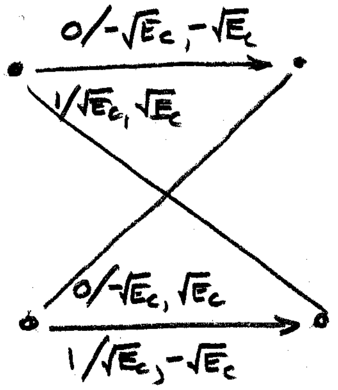
$$\arg\min_{\underline{c}} \left| \underline{r} - \underline{s}(\underline{c}) \right|^2$$

Again, this is a sum of non-negative branch metrics, each a squared Euclidean distance.

$$\mu(\underline{r}(m), \sigma(m-1), \sigma(m)) = \left\| \underline{r}(m) - \underline{s}(\sigma(m-1), \sigma(m)) \right\|^2$$

e.g. for binary antipodal $0 \rightarrow -\sqrt{E_c}$

$1 \rightarrow \sqrt{E_c}$



e.g. for the 1, 0 transition

$$\mu(\underline{r}(m), 1, 0)$$

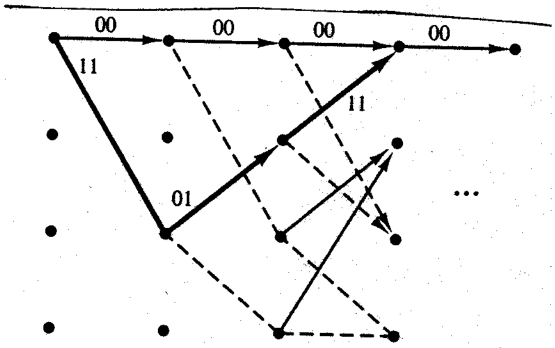
$$= \left(r_1(m) + \sqrt{E_c}\right)^2 + \left(r_2(m) - \sqrt{E_c}\right)^2$$

Since equipower, can simplify this branch metric(1,0) to

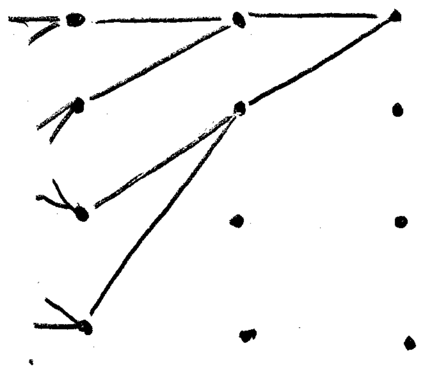
$$\max \ \underline{r}(m) \cdot \underline{s}(\cdots) = r_1(m)\, s_1(m) + r_2(m)\, s_2(m)$$

$$\mu(\underline{r}(m), 0, 1) = r_1(m) - r_2(m)$$

- Viterbi starts in known state and ends known state



expands the number of states each step until 4 states



reduces number of states each step down to 1

- In principle, we release the sole survivor path when we get to the final state. Could be a lengthy delay — can't use first bits until last are received

- In practice, wait 5 or more constraint lengths and release oldest bits from any survivor (or the one with best metric) in a moving window fashion.