8.8 Performance of Convolutional Codes

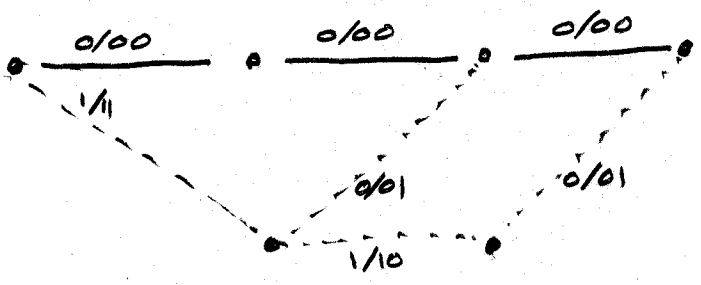
- Convolutional codes are linear. Like linear block codes, it is therefore sufficient to consider the all zero sequence to have been transmitted.

- Also like block codes, we can express WER as

$$P_e' \leq \sum_{d=d_{min}}^{N} v_d P_2(d) \leq \left(2^N - 1\right) P_2(d_{min})$$

but not very useful with such a huge number of alternatives. And, in any case, we wanted the BER

- So focus on error events, prob of an error event of length N, with number of bit errors it causes. Use two state as an example



length 2 : 1 event, 1 error, $d_H = 3$ ← min Hamming dist called d_{free}

length 3 : 1 event, 2 errors, $d_H = 4$

etc.

- This one is easy, because
  - → linear, consider only all zeroes, don't have to average over transmitted sequences
  - → only one event of each length, so don't have an "inner" union bound over all events of each length. Not true of convolutional codes in general. (Also only one event of each d_H).

- The BER is therefore upper bounded by

$$P_b \le \sum_{N=2}^{\infty} e_N P_2(d_{HN}) = \sum_{d=d_{free}}^{\infty} e(d) P_2(d)$$

simplified from

$$\sum_N \sum_{\substack{i \in \text{events} \\ \text{of length } N}} \text{errors}_i \cdot P_{2i}$$

> The "transfer function" simplifies counting events

The rest is similar to block codes in Section 8.5

- Hard decisions (this code only)

$$P_b \le \sum_{d=d_{free}}^{\infty} e(d) P_{2h}(d) \le \sum_{d=3}^{\infty} (d-2) \left(\sum_{i=\lfloor \frac{d+1}{2} \rfloor}^{d} \binom{d}{i} P_c^{\,i} (1-P_c)^{d-i} \right)$$

(P_c: prob error, coded bit; $(d-2)$: for this feeble code)

$$= \sum_{d=3}^{\infty} \sum_{i=\lfloor \frac{d+1}{2} \rfloor}^{d} (d-2) \binom{d}{i} P_c^{\,i} (1-P_c)^{d-i}$$

- For high SNR, $P_c \ll 1$, and the sum is dominated by $i = \lfloor (d+1)/2 \rfloor$ term

$$P_b \approx \sum_{d=3}^{\infty} (d-2)\binom{d}{\lfloor \frac{d+1}{2} \rfloor} P_c^{\lfloor d+1/2 \rfloor}$$

in pairs, odd, next even

no longer a bound

$$= \sum_{\substack{d \geq 3 \\ d \text{ odd}}} (2d-3)\binom{d}{\frac{d+1}{2}} P_c^{(d+1)/2}$$

again drop higher powers

$$\approx 3\binom{3}{2} P_c^2 = 3P_c^2$$

- If binary antipodal:

$$P_c = Q(\sqrt{2\gamma_c}) \leq \frac{1}{2} e^{-\gamma_c}$$

$$P_b \approx \frac{3}{2} e^{-2\gamma_c}, \quad \gamma_c \gg 1$$

- Soft decisions. Again,

$$(d_E)^2 = 4E_c d_H \qquad P_{2s} = Q\left(\frac{d_E/2}{\sqrt{N_0/2}}\right) = Q\left(\sqrt{2\gamma_c d_H}\right)$$

$$\leq \frac{1}{2} e^{-\gamma_c d_H}$$

and

$$P_b \leq \sum_{d=d_{free}}^{\infty} e(d) P_{2s}(d) = \sum_{d=3}^{\infty} (d-2) Q\left(\sqrt{2\gamma_c d}\right)$$

$$\leq \sum_{d=3}^{\infty} (d-2)\frac{1}{2} e^{-\gamma_c d} \approx \frac{1}{2} e^{-3\gamma_c}, \quad \gamma_c \gg 1$$

- Was it worth the effort?
  - uncoded $P_b = Q(\sqrt{2\gamma_b}) \leq \frac{1}{2} e^{-\gamma_b}$
  - coded, rate $\frac{1}{2}$, $\gamma_c = \gamma_b / 2$
    - hard $P_b \approx \frac{3}{2} e^{-2\gamma_c} = \frac{3}{2} e^{-\gamma_b}$

      Poorer than uncoded – but this is a very feeble code, selected as an intro example
    - soft $P_b \approx \frac{1}{2} e^{-3\gamma_c} = \frac{1}{2} e^{-\frac{3}{2}\gamma_b}$

      Better than uncoded.

Good codes (also see Proakis, Digital Communications)

TABLE 10-2 RATE $\frac{1}{2}$ MAXIMUM FREE-DISTANCE CODES

Constraint length L	Generators in octal		d_{free}
3	5	7	5
4	15	17	6
5	23	35	7
6	53	75	8
7	133	171	10
8	247	371	10
9	561	753	12
10	1167	1545	12
11	2335	3661	14
12	4335	5723	15
13	10533	17661	16
14	21675	27123	16

Odenwalder (1970) and Larsen (1973).

TABLE 10-3 RATE $\frac{1}{3}$ MAXIMUM FREE-DISTANCE CODES

Constraint length L	Generators in octal			d_{free}
3	5	7	7	8
4	13	15	17	10
5	25	33	37	12
6	47	53	75	13
7	133	145	175	15
8	225	331	367	16
9	557	663	711	18
10	1117	1365	1633	20
11	2353	2671	3175	22
12	4767	5723	6265	24
13	10533	10675	17661	24
14	21645	35661	37133	26

Odenwalder (1970) and Larsen (1973).

Although we don't know the number of events with distance d_{free} or the number of errors involved, we do have

$$P_{2s}(d_{free}) = Q\left(\sqrt{2\gamma_c d_{free}}\right) = Q\left(\sqrt{2\frac{k}{n} d_{free}\gamma_b}\right)$$

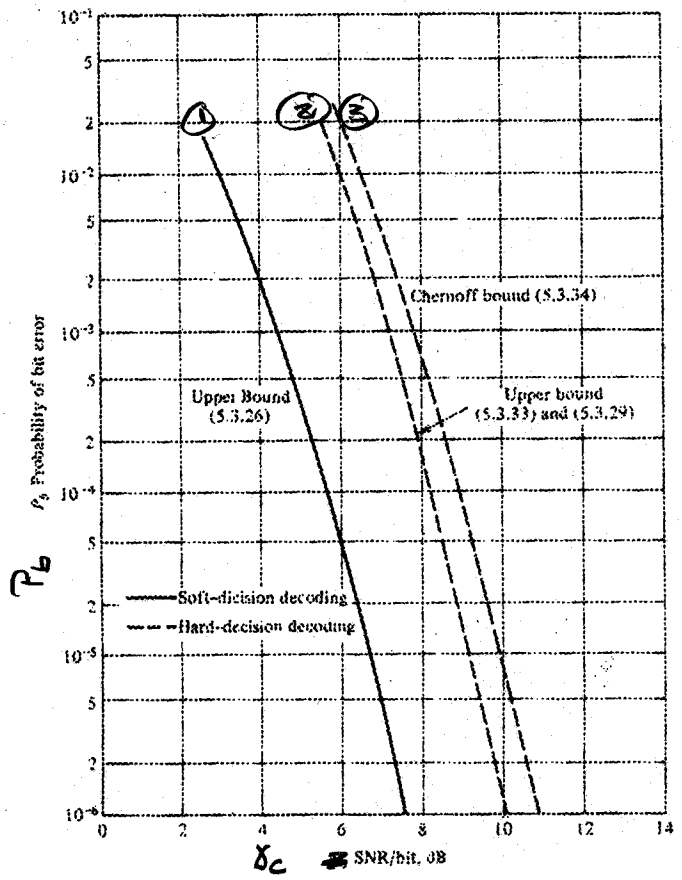
so asymptotic SNR improvement

$$\frac{k}{n} d_{free}.$$

<u>example</u> commonly used rate $\frac{1}{2}$, constraint length 7

from Table 10-2 $d_{free} = 10$

so a boost of $\frac{1}{2} \cdot 10 = 5$ or 7 dB !



This is performance of $n=3$ $k=1$ $L=3$ convolutional code.

Do we get the predicted improvement?

See Table 10-3.

①, ② use the weight distribution

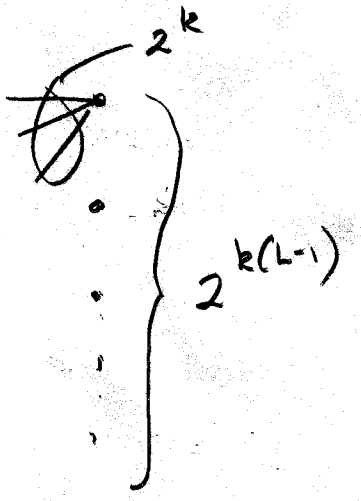
Computational Effort

- Code with n, k, L has (see p 8.7.1)

$$2^{k(L-1)} \text{ states}$$

each with

$$2^{k} \text{ successors and predecessors}$$



At each time step, there are

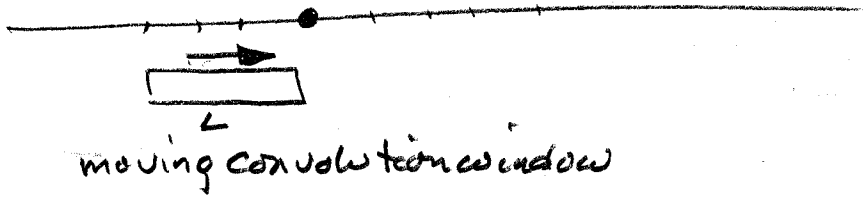
$$2^{k} \cdot 2^{k(L-1)} = 2^{kL}$$

path metrics to evaluate. A lot.

- Several approximate algorithms to reduce computation:
  - M algorithm
  - T algorithm
  - per-survivor processing

Local Effect

- Note that info bits directly affect only output within the constraint length.



- When decoding we (roughly) use the received bits within that constraint length, plus perhaps a constraint length or 2 on either side to reveal identity of neighbouring bits. So although VA in principle uses whole sequence, its "effective memory" is about 3 to 5 constraint lengths — explains safe to release after 5.

- In contrast to block codes, where each bit in principle affects the whole block — need to receive all n bits before decoding any of them.