Tuesday, December 1, 2020 6:02 PM

Hard-decision decoding

1) Demodulate the received signal (using MF and sampling to obtain y vector in $\mathbb{R}^n$)

2) Compare each signal received with constellation points to map to a word $\hat{c} \in \{0,1\}^n$

3) Find the "nearest" codeword $c \in \mathcal{C}$ to $\hat{c}$
↑ minimum Hamming Distance

ex. 1011
1101
$d = 2$

1, 2 — use MF/constellation stuff from a month ago

3 - How do we find "nearest" codeword?

for an (n,k) linear block code

We define the Standard Array as follows:

The first row is

$c_1 \quad c_2 \quad \cdots \quad c_M$

where $\mathcal{C} = \{c_1, \ldots, c_M\}$, $M = 2^k$, $c_1 = 00\ldots0$

each c_i is n-bits.

to determine second row; consider all length-n words not yet represented, and pick one w/ minimum weight e_1 (a choice, not unique)
≠ not like last week's es

row 2 is

$c_1 \oplus e_1 \quad c_2 \oplus e_1 \quad \cdots \quad c_M \oplus e_1$

$c_1 \oplus e_1 = e_1$ as $c_1 = 00\ldots0$

Continue until there are 2^{n-k} rows

$$S = \begin{pmatrix} c_1 & c_2 & \cdots & c_M \\ e_1 & c_2 \oplus e_1 & \cdots & c_M \oplus e_1 \\ e_2 & c_2 \oplus e_2 & \cdots & c_M \oplus e_2 \\ \vdots & & & \\ e_{2^{n-k}-1} & c_2 \oplus e_{2^{n-k}-1} & \cdots & c_M \oplus e_{2^{n-k}-1} \end{pmatrix}$$

THM: all elements of S are distinct

Proof a) same row $c_i \oplus e_k = c_j \oplus e_k$

$\rightarrow c_i = c_j$ only if $i = j$

$\rightarrow c_i = c_j$ only if $i = j$

b) $e_a \oplus c_i = e_b \oplus c_j$

$\rightarrow e_a \oplus (c_i \oplus c_i) = e_b \oplus (c_j \oplus c_i)$

$\rightarrow$

$e_a = e_b \oplus c$ for some $c \in C$

by linearity

if this is the case then they are in the same coset

contradiction

If received $\hat{c}$ is not in C, it is in S

Want to find the error it corresponds to e_i and recover c

sent: 1011

received: 0015

0000 1011 ← correct to right word 1111 0100

0010 1001 ← this was my error 1101 0110

THM If H is the parity check matrix, and z_1, z_2 are elements of the same row of S (the same coset)

then $z_1 H^T = z_2 H^T$

are elements of the same [illegible]

then we have $\boxed{z_1 H^T = z_2 H^T}$

Proof. $z_1 = e_a \oplus c_i$, $z_2 = e_a \oplus c_j$

So $z_1 H^T = e_a H^T \oplus c_i H^T \to 0$ as $c_i, c_j \in C$

$z_2 H^T = e_a H^T \oplus c_j H^T$ —

So each coset can be identified w $e_a H^T$

We define the syndrone of a binary seq. z as $\boxed{S = z H^T}$ tells you which coset you're in

→ don't have to store the whole standard array!

only a list of syndrones

The syndrone array is a $2^{n-k} \times l$ array of length $n-k$ sequences s.t. $S_i = z_i H^T$

length $n-k$ sequences ... $e_2 - e_{2^{n-k}-1}$

where e_i is any element of the i^{th} row of S.

(also keep an array of corr. coset leaders $\begin{pmatrix} 0 \\ e_1 \\ \vdots \\ e_{2^{n-k}-1} \end{pmatrix}$)
and their corresponding syndromes: e_i has syndrome S_{i+1}

we receive $\hat{c}$ may or may not be in $\mathcal{C}$.

either way, $\hat{c}$ is in $S \rightarrow \hat{c}H^T = S$ ← a syndrome

which gives us the "smallest" likely error
and the most likely true codeword

$$\hat{c} \xrightarrow{\hat{c}H^T = S} S \xrightarrow{\text{coset leader}} e_a \xrightarrow{e_a \oplus \hat{c} = c} c, \text{ nearest codeword}$$

this minimizes $d(\hat{c}, c)$

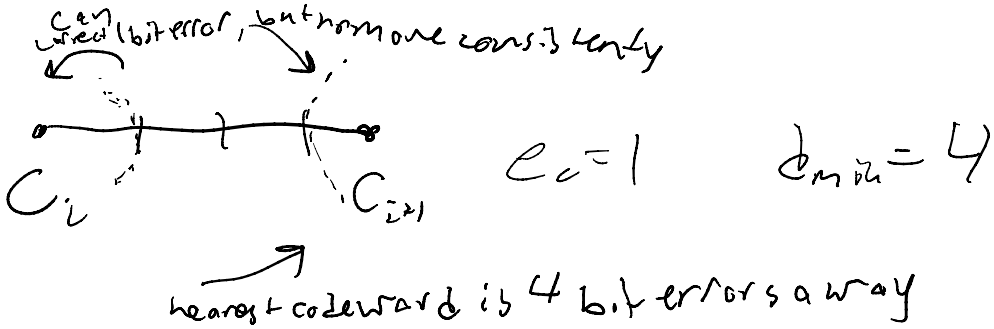
$\hat{c} = c \oplus e \oplus e$

Error Correction

Tuesday, December 1, 2020 6:39 PM

If $\mathcal{C}$ has minimum distance d_{min}, ITX and RX employ hard-decision decoding

$\hat{c} \rightarrow c$ if $d(\hat{c}, c)$ minimum



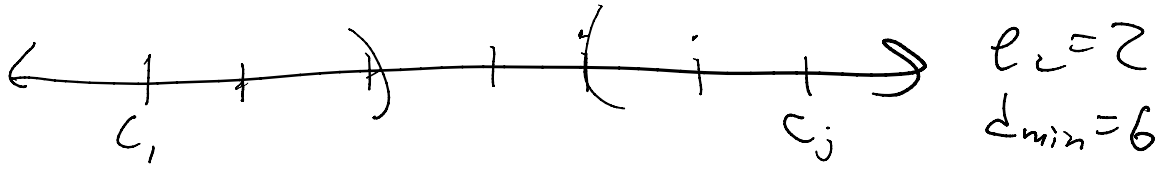
each codeword has a "Hamming Sphere" centered at it containing all correctable errors

Minimum radius is dependent on min. Hamming distance

e_c

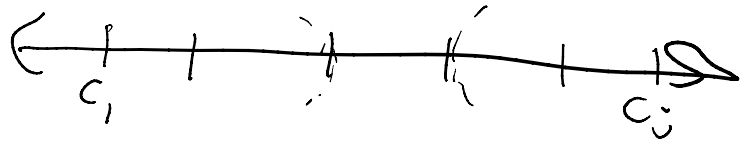
if d_{min} is even (like above)

then $\boxed{d_{min} = 2e_c + 2}$



if d_{min} is odd

$$d_{min} = 2e_c + 1$$



$e_c = 2$

$d_{min} = 5$

for linear block codes using Syndrome method

Burst Error

Tuesday, December 1, 2020 6:48 PM

- A sequence of errors one after the other as a result of an impulse

Before: (BSC) 1011011101111

1001011100

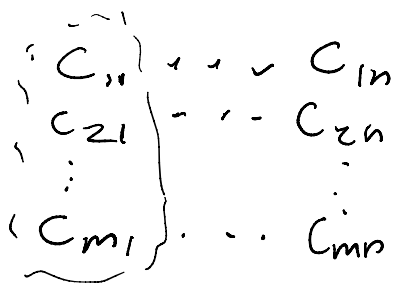
Now: (burst channel) 1011011101111

1011100101111

Not great for our current method:
in BSC, each codeword will have a small # of errors
in burst, most codewords will have 0 errors, but some will have many

Solution: An interleaver of depth m takes m codewords (each length n) and arranges them in rows,

in rows,



usually would TX row-by-row

here TX column by column

RX bits of n codewords at a time, reconstruct array "de-interleave" to receive codewords

usually: $c_{11} c_{12} \cdots c_{1n} c_{21} c_{22} \cdots c_{2n} \cdots$

how: $c_{11} c_{21} c_{31} \cdots c_{m1} c_{12} c_{22} \cdots$

Burst in first case: ruins a codeword

Burst in second case: flips single bits from many codewords

Say a burst of size m occurs, then

in first case m bits of one codeword are flipped

in second case, 1 bit of every codeword is flipped

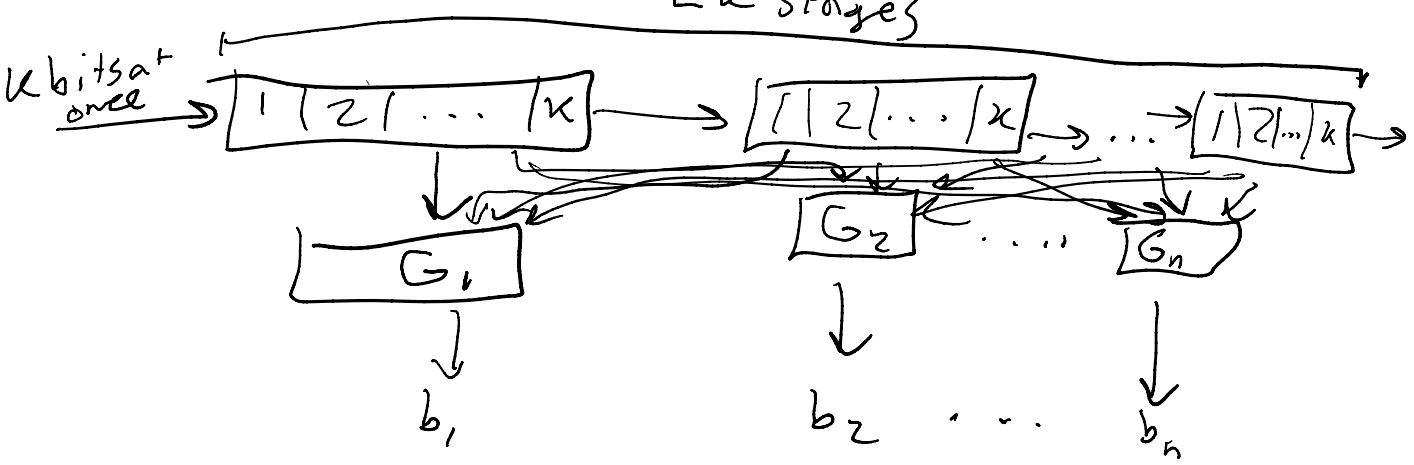
→ results in no loss if $d_{min} \geq 3$

Convolutional Codes

Tuesday, December 1, 2020 7:07 PM

- Unlike LBCs convolutional codes have memory
- widely used because of error correction capabilities

framework: we map k-bits sequences to n-bit sequences, now non-stationarily — we use the following shift register circuit



L past/present input data sequences affect the output codeword

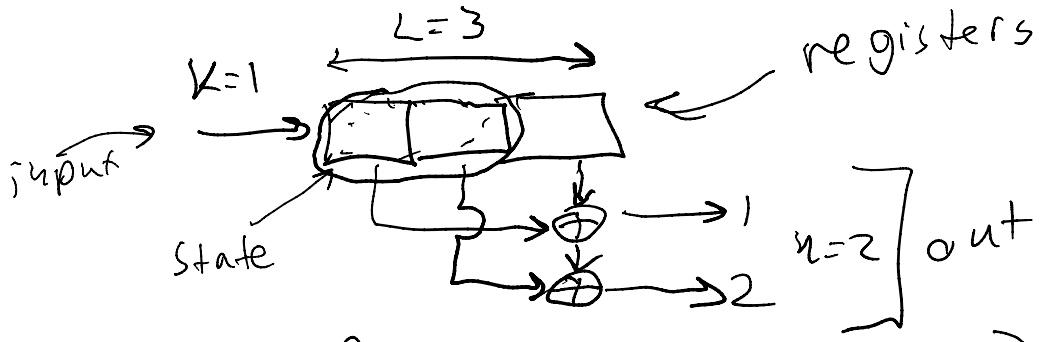
$L \equiv$ constraint length

G_i is some linear comb. of the Lk present bits

rate of the code: $R = k/n$ (same as before)

EX a rate 1/2 encoder, w/ $L = 3$ shown below

EX a rate 1/2 encoder, w/ L=3 shown below



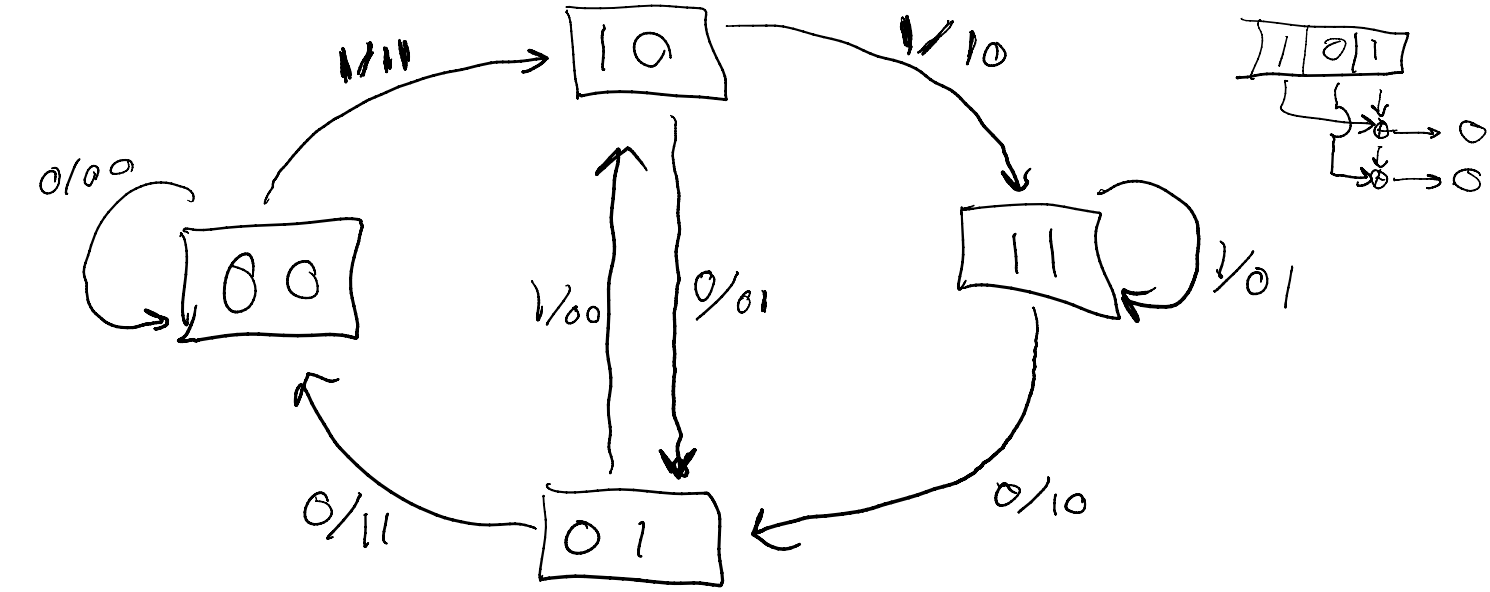
This is a finite state machine w/ $2^{(L-1)\cdot K}$ states

$2^{2-1} = 4$ states

The first $(L-1)k$ bits determine the state

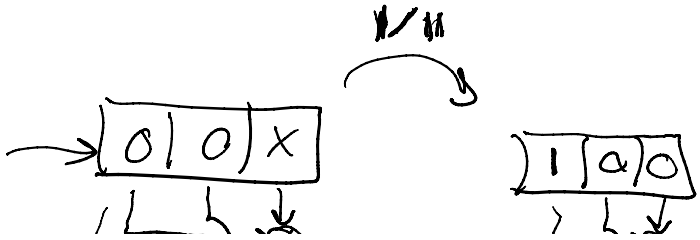
Each input controls the transition

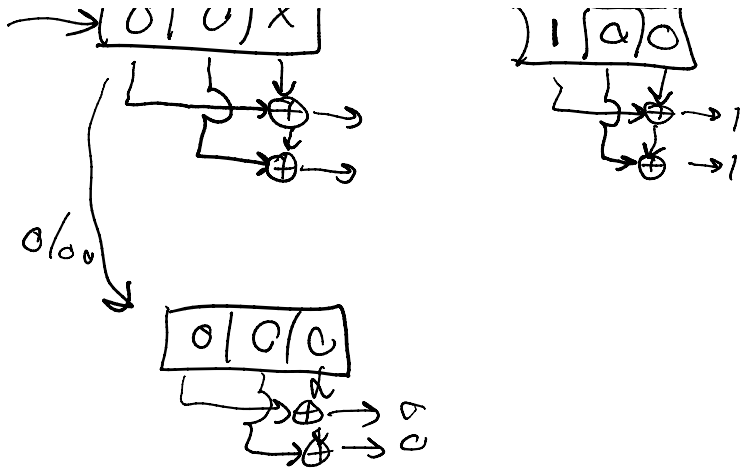
Draw a state transition diagram



state $\xrightarrow{\text{in/out}}$ state

after the input





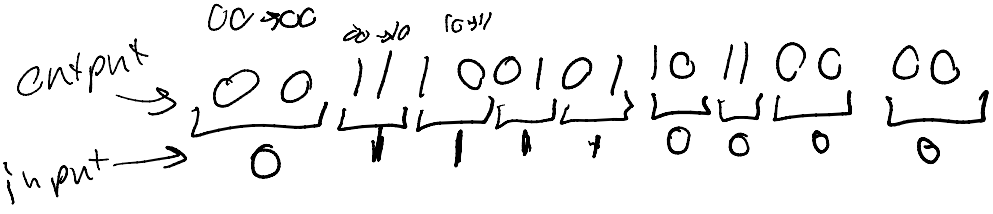
The sequence "11" alone is impossible to decode

need a full sequence

and

knowledge of initial conditions

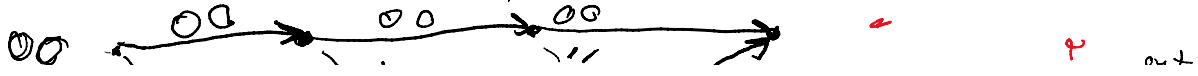
If I know 0 bit-errors, started at 00

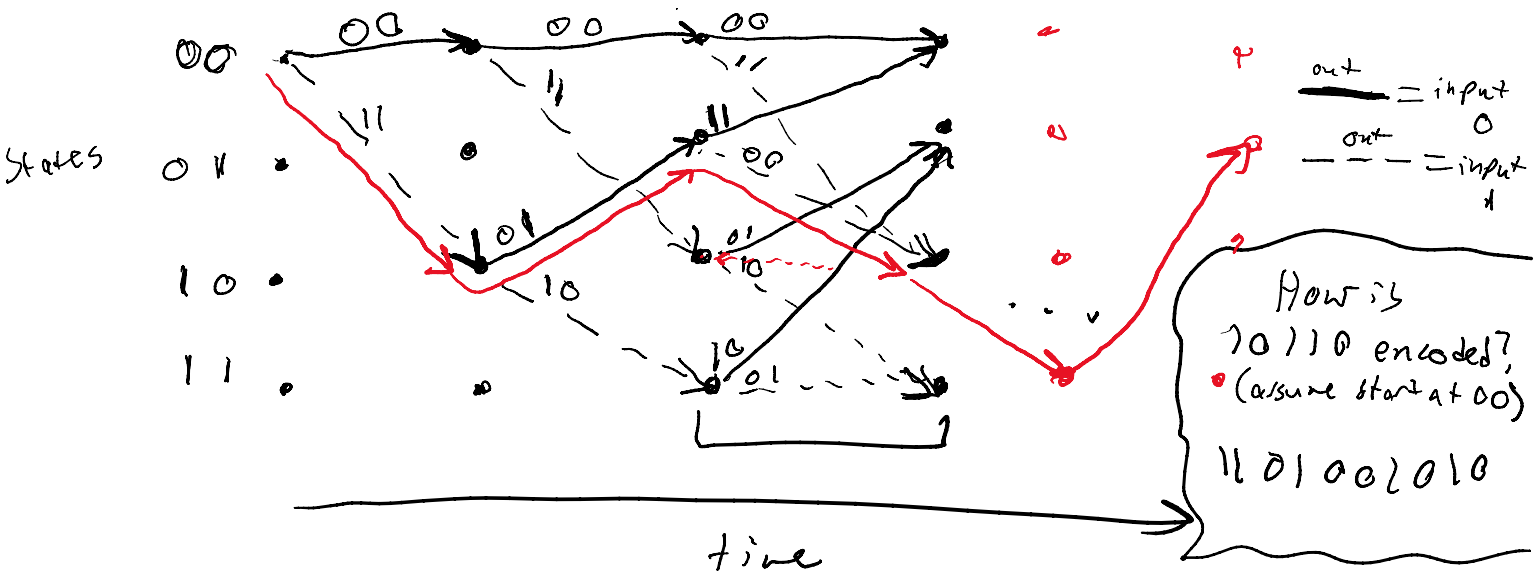


New visual rep. of FSM

Trellis Diagram

Shows transitions in time. 2-D → time-axis vs. state-axis

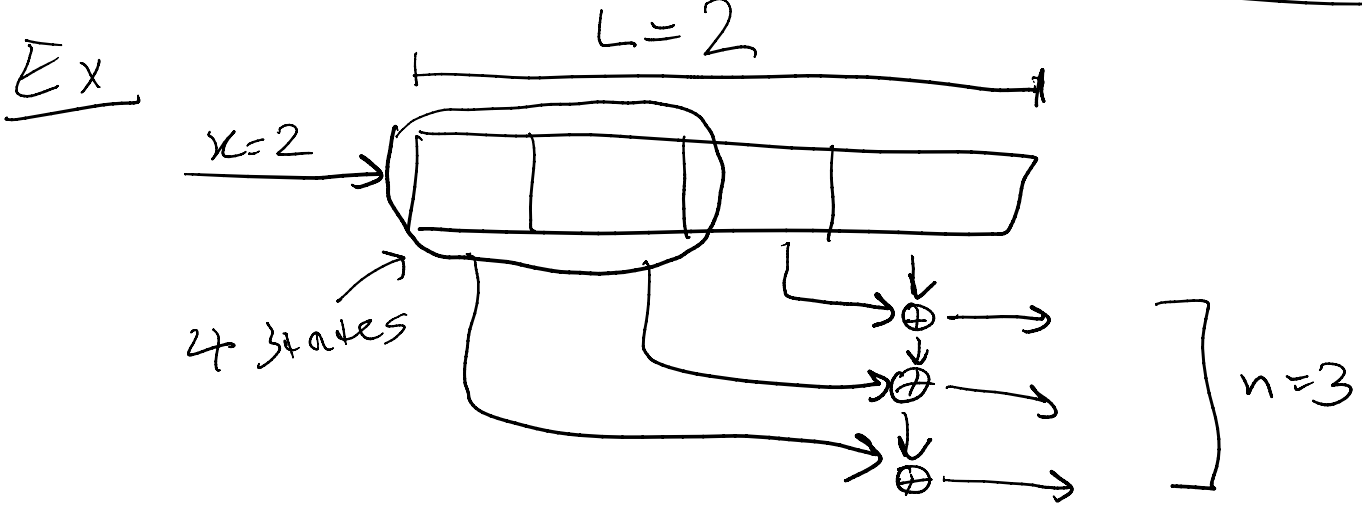




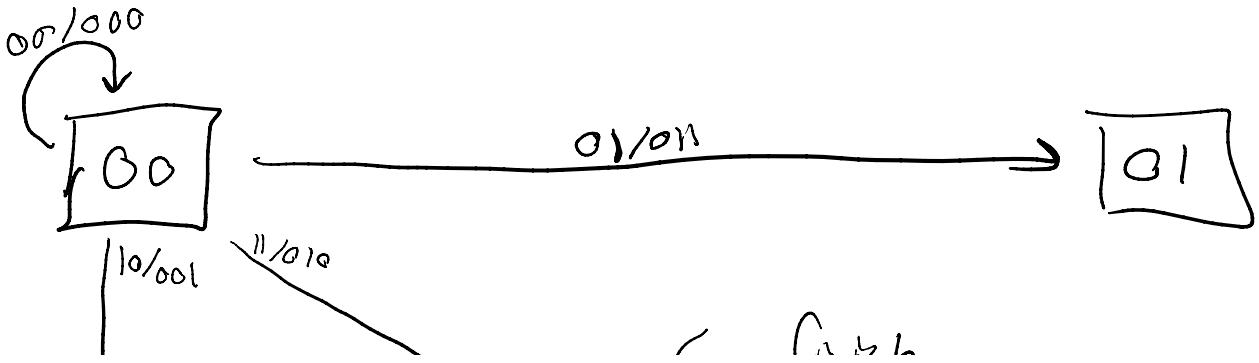
Trellis gives all possible sequences of transitions, and corresponding inputs/outputs

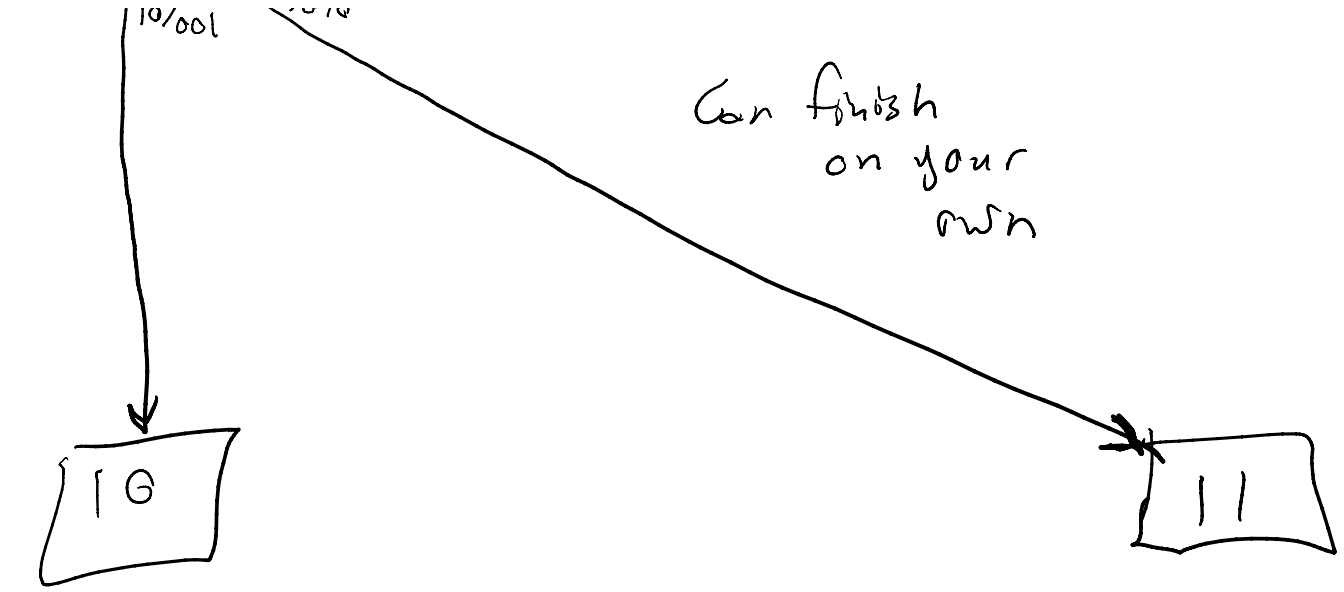
---

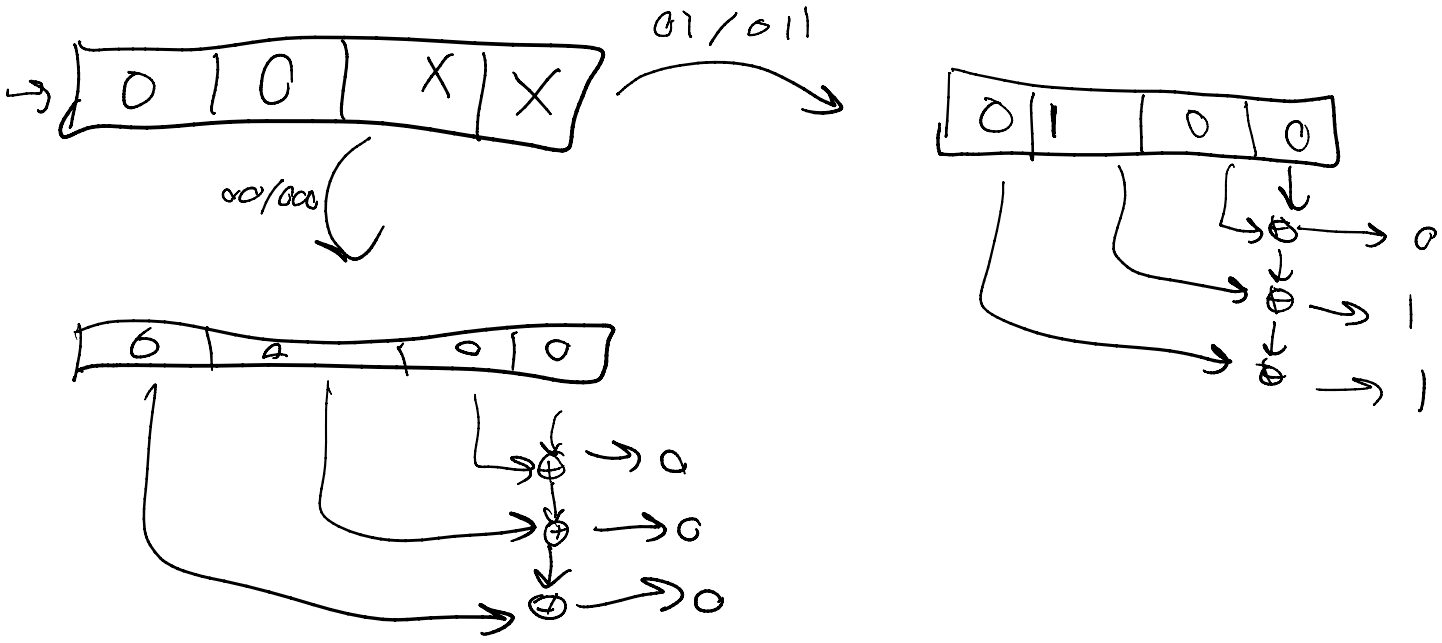
Ex



4 ways to transition from each state




10/001
Can finish
on your
own
10
11


0 0 X X
01/011
00/000
0 1 0 0
0
1
1
0 a 0 0
a
0
0

Viterbi Algorithm

Tuesday, December 1, 2020 8:00 PM

Maximum Likelihood Sequence Estimation (MLSE)

on any FSM

⇒ have noisy outputs + noise and initial conditions

→ I'll guess the sequence of inputs

(note MAPSE is used instead in 4G LTE)

Frame the problem mathematically:

Say we have FSM w/ states

$$S = \{\sigma^0, \sigma^1, \ldots, \sigma^{|S|-1}\}$$

$|S|$ possible states, we know (w log) initial state is σ^0

We feed an input at each time instance m which is a K-bit sequence $\vec{x}(m)$

Denote state at time m as $\sigma(m) \in S$

a sequence of N inputs: $\mathbf{X} = (\vec{x}(1), \ldots, \vec{x}(N))$

Corresponding to a sequence of states;

$$\vec{\sigma} = \{\sigma^0, \sigma(1), \ldots, \sigma(N)\}$$ Length $N+1$

and σ^0

If we know the FSM, then $\mathbf{X}$ totally determines $\vec{\sigma}$

we also assume that $\vec{\sigma}$ uniquely determines $\mathbf{X}$

(in every conv. coder used in practice this is true)

$\mathbf{X} \longleftrightarrow \vec{\sigma}$ contain same info

The FSM output (the encoded signal)

is given by an n-bit seq. at time m $\vec{y}(m)$

$$\mathbf{y} = (\vec{y}(1), \ldots, \vec{y}(N))$$

this is transmitted, corrupted by noise, receive

$$\mathbf{r} = (\vec{r}(1), \ldots, \vec{r}(N))$$

↑ corrupted $\mathbf{y}$

... upred $\mathbf{y}$

we want to recover $\mathbf{x}$ from $\mathbf{r}$

or $\vec{\sigma}$

to perform MLSE, we want to find the $\vec{\sigma}$ which

maximizes $P[\mathbf{r}|\vec{\sigma}]$ ML

maximizes: estimation; $\vec{\sigma}$: sequence

and hopefully $\hat{\vec{\sigma}} = \vec{\sigma}$

$$\text{so } \hat{\vec{\sigma}} = \underset{\vec{\sigma}}{\operatorname{argmax}} P[\mathbf{r}|\vec{\sigma}]$$

$\vec{\sigma}$ specifies a path along the trellis

error occurs here

affects every guess after!

a single unaccounted for error could ruin everything!

Language

we call each state transition $[\sigma(m-1), \sigma(m)]$ a branch

corresponds to 1 input / 1 output

If two paths are the same for time $t \leq m$ but $\vec{\sigma}_1(m+1) \neq \vec{\sigma}_2(m+1)$, we say the paths diverge at $m+1$

similarly, if same after m, then converge at m

For most channels we discussed, errors are time-independent (burst is the exception)

$$\rightarrow \quad P[\mathbf{r} | \vec{\sigma}] = \prod_{m=1}^{N} P[\vec{r}(m) | [\sigma(m-1), \sigma(m)]]$$

from assumption of time-indep of error.

Given state $\sigma(m)$, can transition to 2^k possible $\sigma(m+1)$ values

$\sigma(m+1)$ values

we define the Branch Metric

$$BR_m([\sigma^i, \sigma^j]) = \log P[\vec{r}(m) \mid \sigma(m-1)=\sigma^i, \sigma(m)=\sigma^j]$$

and the Path Metric

$$\boxed{M(\vec{\sigma}) = \sum_{m=1}^{N} BR_m([\sigma(m-1), \sigma(m)])}$$

Branch metric / Path metric < 0, higher prob → nearer to 0

Want to maximize

MLSE is max the $\prod P[\vec{r}(m) \mid [\sigma(m-1), \sigma(m)]]$

is max $\log \prod P[\vec{r}(m) \mid [\sigma(m-1), \sigma(m)]]$

is max $\sum \log P[\vec{r}(m) \mid [\sigma(m-1), \sigma(m)]]$

$= M(\vec{\sigma})$

So MLSE is eq. to finding

$$\boxed{\hat{\alpha} = \underset{\vec{\sigma}}{\operatorname{argmax}}\, M(\vec{\sigma})}$$

$$\hat{\sigma} = \arg\max_{\vec{\sigma}} J^{ML}(\vec{\sigma})$$

---

to find BR_m — need to know the channel

guess: channel is BSC, prob of bit-error is p

$P[\vec{r}|\vec{y}] = p^d (1-p)^{n-d}$ where d is Hamming dist bet y, r

so $\log P[\vec{r}|\vec{y}] = \underbrace{n \log(1-p)}_{\text{depends only on channel; can't minimize/maximize}} + \underbrace{d \log\left(\frac{p}{1-p}\right)}_{\text{interesting term}}$ ← fixed

so $\log P[\vec{r}|\vec{y}]$ is effectively proportional to d the Hamming distance

$$BR^1 = -d(\vec{r}, \vec{y})$$

→ from above, ML is just minimizing Hamming distance

(and in soft-decision, $BR^1 = -d_{\text{euclidean}}(\vec{r}, \vec{y})$)

inconvenient annoyance:

$$BR_m' \equiv d(r(m), y(m))$$

$$M' \equiv \sum_{m=1}^{N} d(r(m), y(m))$$

now want to <u>minimize</u> M'

So $MLSE \equiv \hat{\vec{\sigma}} = \underset{\vec{\sigma}}{\operatorname{argmin}}\, M'(\vec{\sigma}) = \underset{\vec{\sigma}}{\operatorname{argmin}} \sum d$

<u>just minimizing</u> <u>Hamming dist</u> <u>over a whole path</u>

how we know what is going on in Viterbi

but it seems undoable — check every path?

<u>no!</u>

<u>THM</u> Suppose two paths $\vec{\sigma}_1$ and $\vec{\sigma}_2$ diverge at $m+1$ and converge (merge) again at $m+l$ (so $\vec{\sigma}_1(a) = \vec{\sigma}_2(a)$ $\begin{matrix} a \le m \text{ or} \\ a \ge m+l \end{matrix}$)

Let P_0 be the <u>subpath</u> that is common to $\vec{\sigma}_1$ and $\vec{\sigma}_2$

... o be the subpath that is common to σ_1 and $\vec{\sigma}_2$ (undefined for $m+1 \le a \le m+l$), let A_1 and A_2 be the respective subpaths for $\vec{\sigma}_1$ and $\vec{\sigma}_2$ for the $m < a < m+l$

Then $M(\vec{\sigma}_1)$ is better than $M(\vec{\sigma}_2)$

iff

$M(A_1)$ is better than $M(A_2)$

Why do I care? There are N time steps, 2^k possible transitions at each step, so 2^{kN} possible paths! intractable

But there will be some best path (which we don't know)

$\hat{\sigma}$

Say $\hat{\sigma}(a) = \sigma^i$, there are $2^{k(a-1)}$ paths that start at σ^0 and wind up at σ^i at time a

but of all of these the path which is a subpath of $\hat{\sigma}$ is "the best" by above

a priori, we don't know $\hat{\sigma}(a)$! Some have to keep track of at least one path for each state at each time, but that's much less than 2^k.

Then, at N, compare all $|S|$ paths and pick one w/ best M