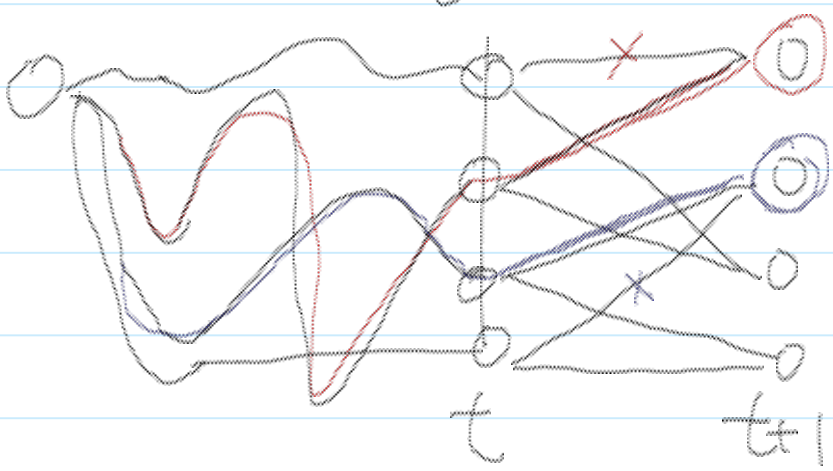
L25

Monday, November 14, 2022 1:39 PM

* The Viterbi algorithm is then described as follows. (A high-level description)

① Intermediate goal: keep track of the maximizing partial path from the origin to each state in the t-th stage.

② For the $(t+1)$-th stage, update the new maximizing partial path for each state, based on the results from the t-th stage → Forward iteration



③ In the final stage, compare the maximizing complete paths for individual

states & select the globally maximizing path.

Detailed implementation of the Viterbi algorithm

① Compute the partial objective function for the given observation y_{2t-1}, y_{2t} for each t

See many previous examples

& we denote the metrics for each segment as m_s, s'

② Compute the forward metric $a_s[t]$.

For the zero-th stage ($t=0$) set the metric $a_{00}[t] = 1$ & $a_s[t] = 0$ for all other states $s \neq 00$. In our running example, we have

$a_{00}[0] = 1, \quad 0 = a_{01}[0] = a_{10}[0] = a_{11}[0]$

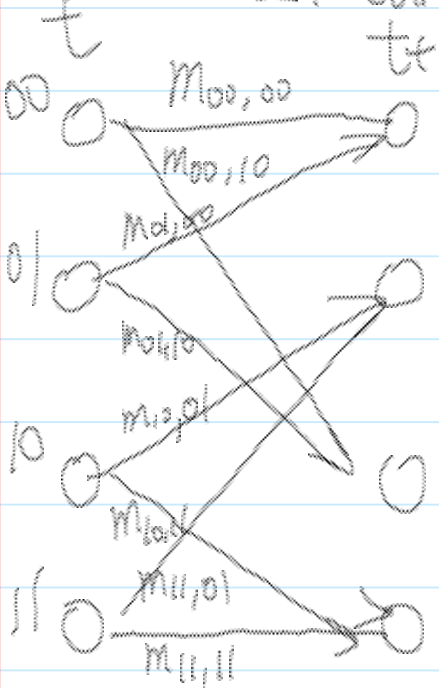
$a_S[t]$ is the metric of the maximizing partial path from the origin to S_t

∵ The origin starts from $S=00$

③ Update the $a_S[t+1]$ based on the following formula

$$a_S[t+1] = \max_{S' \text{ that goes to } S} \{ a_{S'}[t] m_{S',S} \}$$

In our example

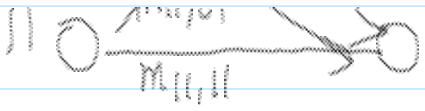


$a_{00}[t+1] = \max(a_{00}[t] m_{00,00}, a_{01}[t] m_{01,00})$

$a_{01}[t+1] = \max(a_{10}[t] m_{10,01}, a_{11}[t] m_{11,01})$

⋮

$a_{11}[t+1] = \max(a_{10}[t] m_{10,11}, a_{11}[t] m_{11,11})$



$a_{11}[t+1] = \max(a_{10}[t]m_{10,11}, a_{11}[t]m_{11,11})$

④ After we update all $a_s[t]$, $t=0,\cdots,T$.

the maximum value

$\max_R \prod_{t=1}^{T} f_t(\overline{P[t]}) = \max_S a_S[T]$, which

is the maximum likelihood value of any codeword

The above procedure describes how to find $\max_R \prod_{t=1}^{T} f_t(\overline{P[t]})$. But we are more interested in

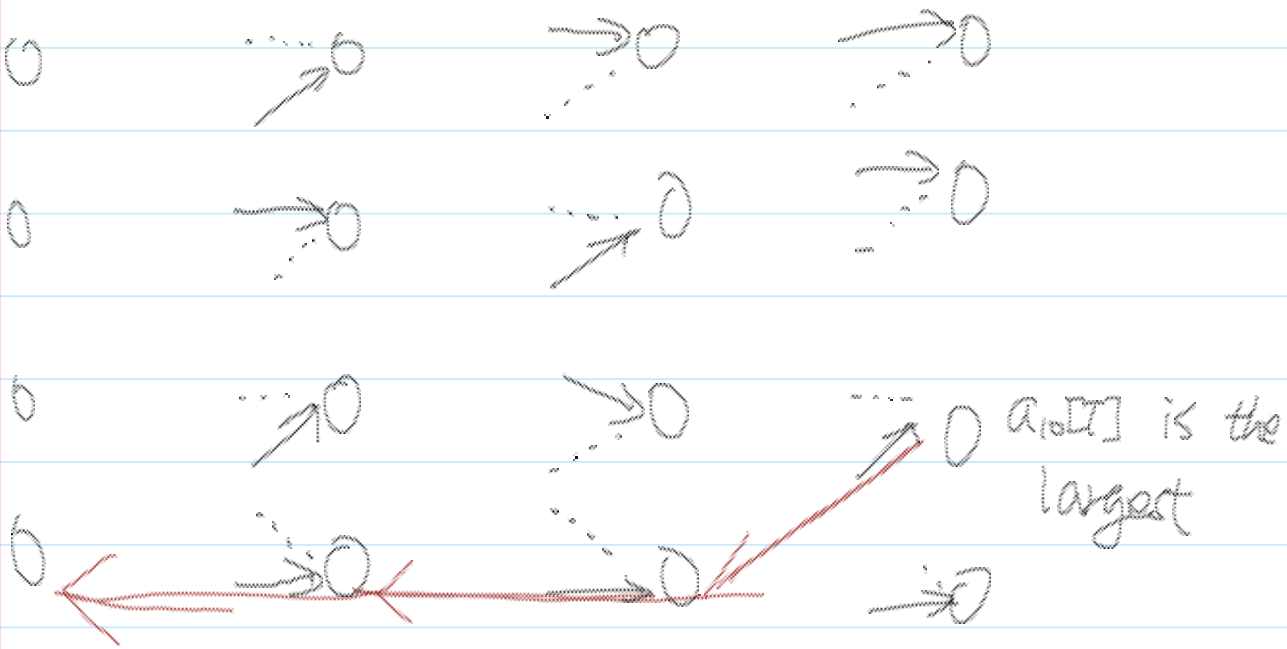
$$\operatorname{argmax} \prod_{t=1}^{T} f_t(\overline{P[t]}).$$

which tells us the most likely codeword

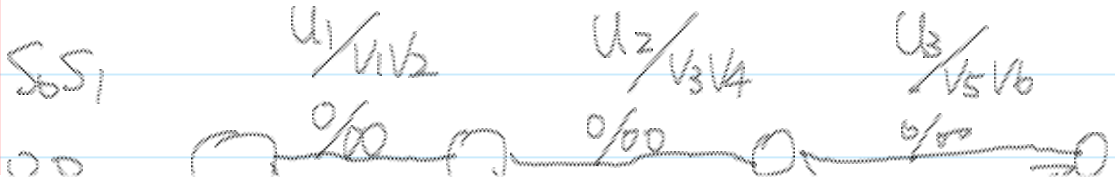
Q: How to find the most likely codeword?

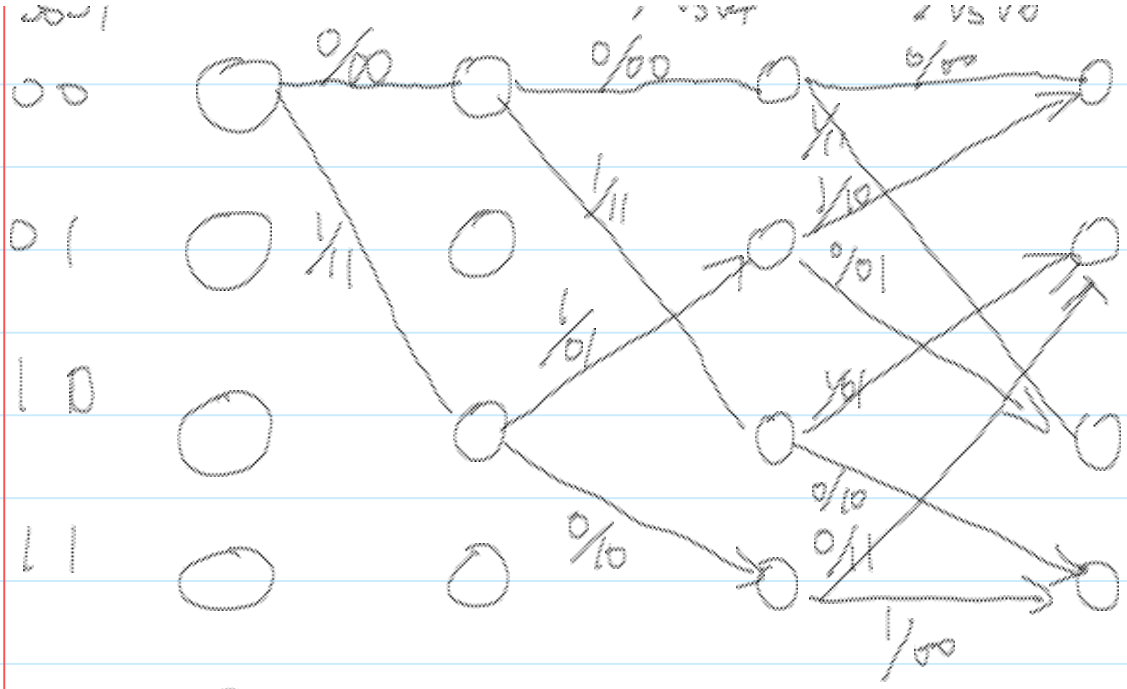
Ans: Similar to the Dijkstra algorithm.

We memorize the choices of the max operations and then trace it backward from the destination back to the origin.

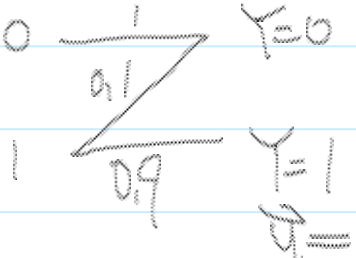


An illustrating example based on the running example code, T=3





Channel: Z-channel

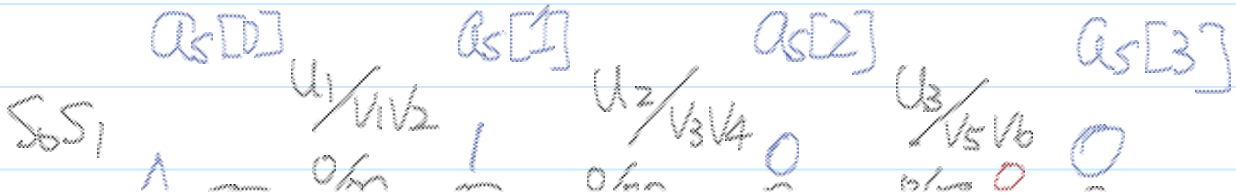


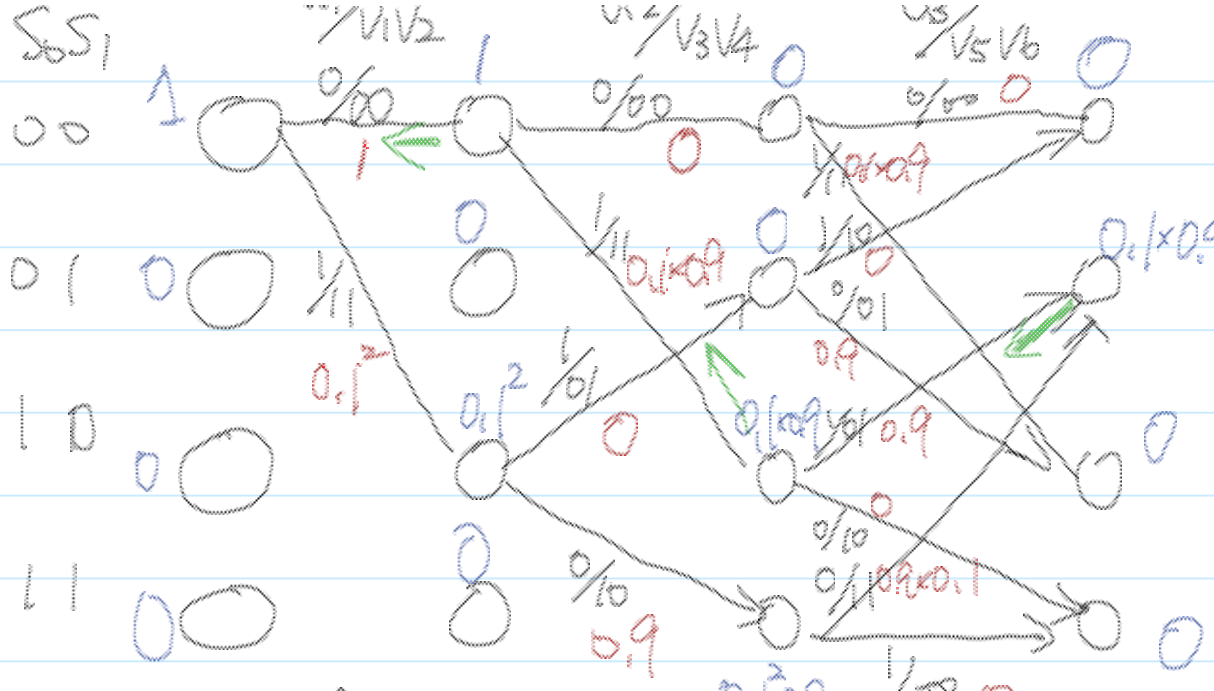
The observation is $\underline{y} =$ 00 10 01

Q: Use the VA to find the optimal codeword

Ans: ① Initialize the partial objective function $f_t(\cdot)$ (or the metric for each segment).

Observation: 00 10 01





② Compute the forward metrics

$$\alpha_{00}[3] = \max(0\times 0,\ 0\times 0) = 0$$

$$\alpha_{01}[3] = \max(0.1\times 0.9\times 0.9,\ 0.1^2\times 0.9\times 0.1\times 0.9)$$

$$= 0.1\times 0.9^2$$

$$\alpha_{10}[3] = \max(0\times 0.1\times 0.9,\ 0\times 0.9) = 0$$

$$\alpha_{11}[3] = \max(0.1\times 0.9\times 0,\ 0.1^2\times 0.9\times 0) = 0$$

③ Backward trace

The most likely codeword

$\hat{V} = 00\ 11\ 01$

Summary of convolutional codes:

① Encoding: the info bits steer the path

② Decoding: Iteratively compute the maximizing partial paths & then trace it backward to find p^*

③ The use of the path metric $f_t(\cdot)$ is an abstraction of the prob quantities. There is no need to worry about the prob meaning of $a_s[t]$

Several variants of the Viterbi decoder

① Multiplication is hard, addition is easy

$$\max_p \prod_{t=1}^{T} f_t(\overline{p[t]}) \equiv \max_p \sum_{t=1}^{T} \log f_t(\overline{p[t]})$$

We only need to change the metrics used in the VA. & the corresponding update rule

$$a'_{00}[0] = \log(a_s[0]) = \log(1) = 0, \quad a'_s[0] = \log(0) \text{ for } s \neq 00.$$

$$m'_{s',s} = \log(m_{s',s}) = \log\left(P_{y_t|x_t(v_{t-1},v_t)}\right)$$

$$a'_s[t+1] = \max_{s'} \left\{ a'_{s'}[t] + m'_{s',s} \right\}$$

s' that reaches s

The new VA maximizes

$$\max_{p} \sum_{t=1}^{I} \log f_t(\vec{p}[t])$$

(2) Many people prefer minimization over maximization

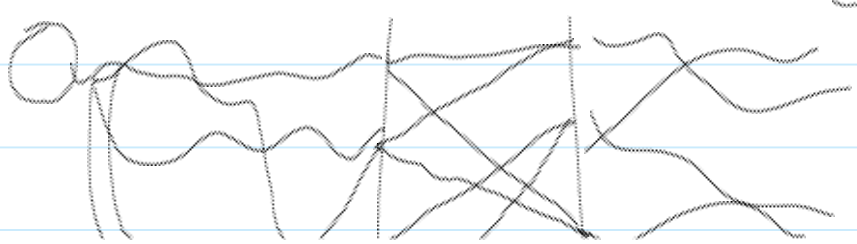
$$\Rightarrow m''_{s',s} = -m'_{s',s} = -\log\left(P_{y_{2t-1},y_{2t}|v_{t-1},v_t}\right)$$

$$a''_s[0] = \begin{cases} 0 & s=0 \\ \infty & \text{otherwise} \end{cases}$$

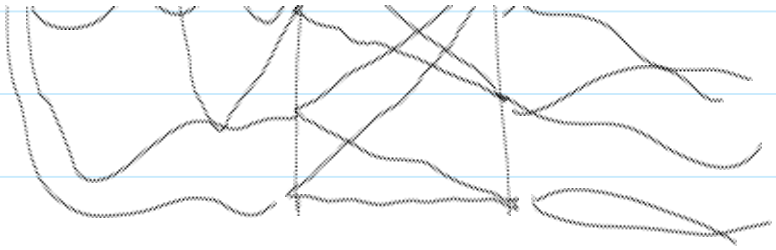
$$a''_s[t+1] = \min_{s' \text{ that reaches } s} \left\{ a''_{s'}[t] + m''_{s',s} \right\}$$

(3) VA: One-way parsing the trellis structure.

Two-way VA: Parse the trellis from both ends. (Trading the hardware complexity for decoding speed.)



$\frac{1}{2}$ the



$\frac{1}{2}$ the decoding time

Forward direction

$\alpha_{00}[0] = 1 \quad \alpha_S[0] = 0 \quad \forall \; S \neq 00$

$$\alpha_S[t+1] = \max_{S' \text{ that reaches } S} \{ \alpha_{S'}[t] \cdot m_{S',S} \}$$

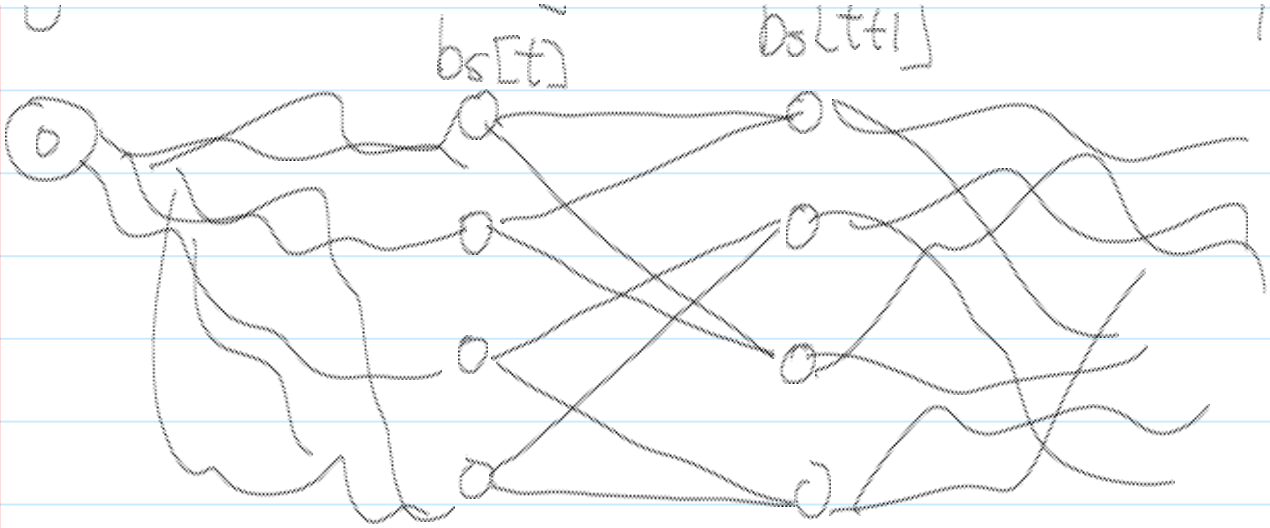
Backward direction

$b_S[T] = 1 \quad$ for $S = 00, 01, 10, 11$

$$b_S[t-1] = \max_{S' \text{ that is reachable by } S} \{ b_{S'}[t] \cdot m_{S,S'} \}$$

When the forward & backward computations meet

0 $\quad \alpha_S[t]$ $\quad b_S[t]$ $\quad b_S[t+1]$ $\quad$ 7

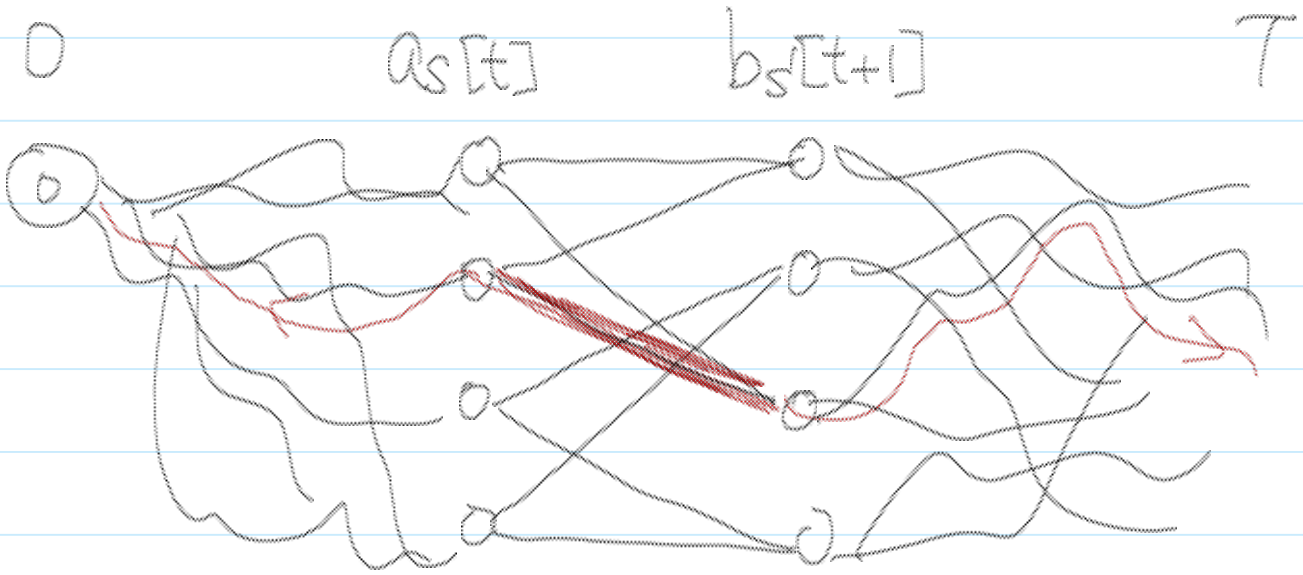


$$\max_{\vec{p}} \prod_{t=1}^{I} f_t(\vec{p}[t]) = \max_{s} \underbrace{\{ a_s[t]\, b_s[t] \}}_{\text{total} \leq |\text{States}| \text{ terms.}}$$

Q: How to find the $\underset{\vec{p}}{\text{argmax}} \prod_{t=1}^{I} f_t(\vec{p}[t])$

A: Record the choices of the max operations during the computation of $a_s[t+1]$ & $b_s[t-1]$

Then trace the path both forward & backward.



④ VA with delay constraints
(Window-based VA)

One drawback of the VA is that it needs to parse the entire trellis structure, ⇒ Need to wait for all $y_1, y_2 \cdots, y_n$ to arrive before decoding ⇒ longer delay

One solution is to sacrifice the

performance by focusing on some window

Use the data we have had until t+W to decide the best partial path, & the best V_{t-1}, V_{st} values. Repeat it for different t.

Generally the window size is chosen as

5 × (the number of delay units)

(5) Use the VA as a <u>minimum distance decoder</u>

$$\text{Viterbi} \equiv \max_{R} \prod_{t=1}^{I} f_t(P[t])$$

$$= \min_{\bar{p}} \sum_{t=1}^{T} h_t(\bar{p}[t])$$

where $h_t(\cdot) = -\log(f_t(\cdot))$

We can choose $h_t(\cdot)$ to be the distance function instead of the likelihood

$$h_t(\cdot) = |\bar{v}[t] - \bar{y}_{obs}[t]|$$

The VA thus becomes a minimum distance decoder.

Example: ① For binary symmetric channels.

$h_t(\cdot)$ is the Hamming distance

② For binary-input Laplacian channels

$h_t(\cdot)$ is the Euclidean distance

③ For binary-input white Gsn channels.

$h(t) = (x - y)^2$ is the square error.

Variants: ① product $\xrightarrow{\log}$ sum ② minimization ③ 2-way VA. ④ Window-based scheme ⑤ Use it as a