L26

Monday, November 14, 2022 1:40 PM

* The VA is an efficient codeword-based ML decoder. 6 different variants.

Q: Can we also design an efficient bit-based ML decoder?

Recall that

$$\prod_{t=1}^{T} f_t(\vec{p}[t]) = P(\vec{Y}=y \mid \vec{V}=\vec{p}) = \frac{P(\vec{Y}=u, \vec{V}=\vec{p})}{2^{-\#\text{ u bits}}}$$

A codeword-based decoder finds

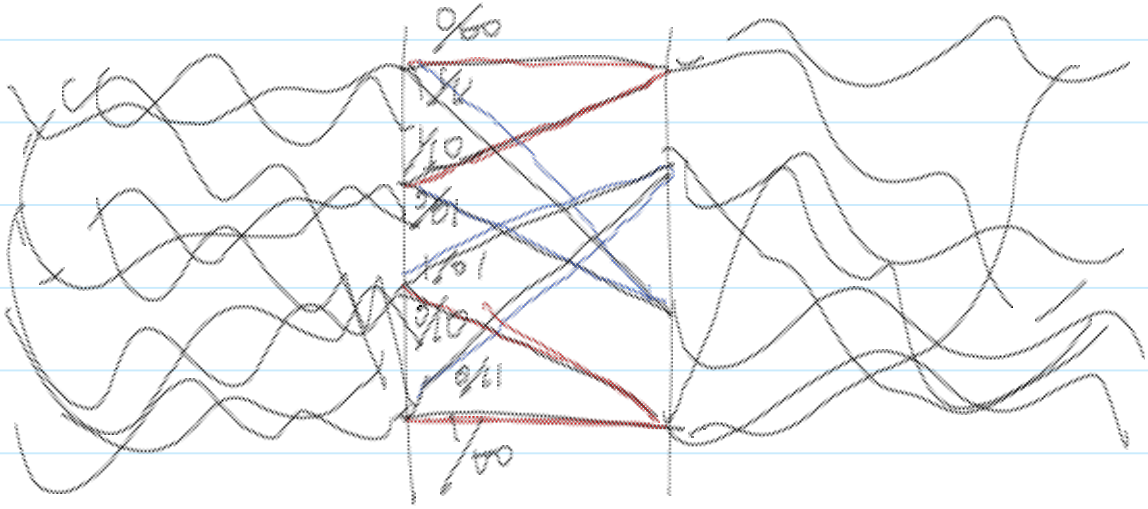
$$\underset{\vec{p}}{\operatorname{argmax}} \prod_{t=1}^{T} f_t(\vec{p}[t])$$

A bit-based decoder of the ith bit finds

$$\operatorname{argmax} \left(\sum \prod_{t=1}^{T} f_t(\vec{p}[t]) \right)$$

$$\underset{x=0,1}{\operatorname{argmax}} \left(\sum_{\vec{p}:\, p[i]=x} \prod_{t=1}^{t} f_t(p[t]) \right)$$

Illustration



If we are interested in

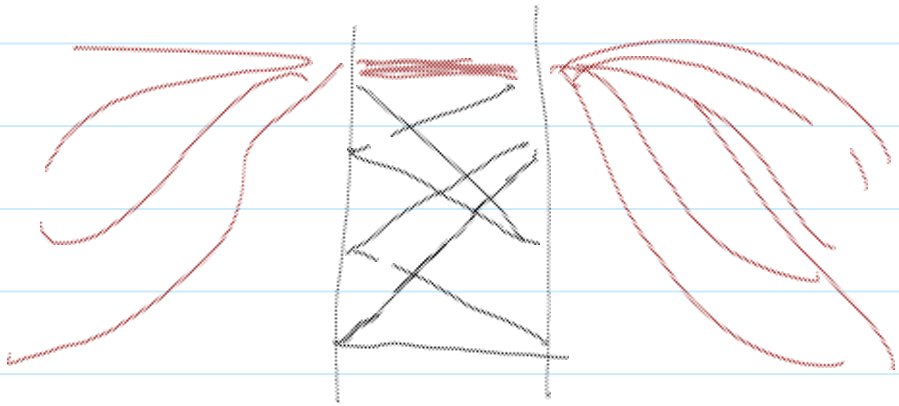
$v_t = 0$ vs. $v_t = 1$

$$\sum_{\text{all paths using}} \prod_{t=1} f_t(p[t]) \quad \text{vs.} \quad \sum_{\text{all paths using}} \prod_{t=1} f_t(p[t])$$

How to compute the sum efficiently?

We first look at each segment respectively.

Sum of



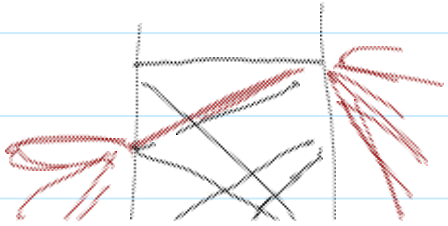
$$= \left(\sum_{\text{○ ○ / ○ ○ / ○ ○}} \prod_{s=1}^{t-1} f_s(\overline{p[s]}) \right) f_t\left(\begin{smallmatrix} \circ & \circ \\ \circ & \circ \end{smallmatrix}\right)$$

A generalized form of the distributive rule

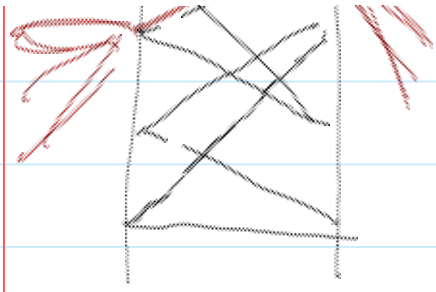
$$a_1 b c_1 + a_2 b c_2 + a_1 b c_2 + a_2 b c_1 = (a_1 + a_2) b (c_1 + c_2)$$

$$\left(\sum_{\text{○ ○ / ○ ○ / ○ ○}} \prod_{s=1}^{t-1} f_s(\overline{p[s]}) \right)$$

Similarly $=$



$$\left(\sum_{\text{○ ○ / ○ ○ / ○ ○}} \prod_{s=1}^{t-1} f_s(\overline{p[s]}) \right) f_t\left(\begin{smallmatrix} \circ & \circ \\ \circ & \circ \end{smallmatrix}\right)$$



$$\left(\sum_{\text{all } p[1..t]} \prod_{s=t+1}^{T} f_s(p[s]) \right)$$

In summary, if we know

$$\left(\sum_{\text{all } p[1..t]} \prod_{s=1}^{t-1} f_s(p[s]) \right), \left(\sum_{\text{all } p[1..t]} \prod_{s=1}^{t-1} \cdot f_s(p[s]) \right) \ldots$$

&

$$\left(\sum_{\text{all } p[1..t]} \prod_{s=t+1}^{T} f_s(p[s]) \right) \left(\sum_{\text{all } p[1..t]} \prod_{s=1}^{t} f_s(p[s]) \right) \ldots$$

then we can compute

then we can compute

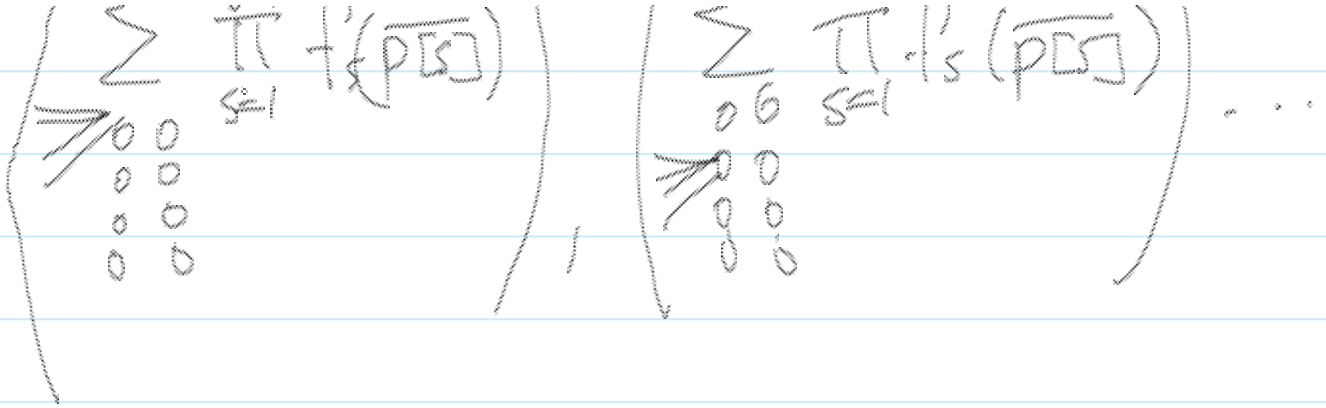
$$\sum_{\text{all paths using } \nearrow \searrow} \prod_{t=1}^{I} f_t(p[t]) \quad \text{vs.} \quad \sum_{\text{all paths using } \nearrow \not\searrow} \prod_{t=1}^{I} f_t(p[t])$$

efficiently. Q: how to find

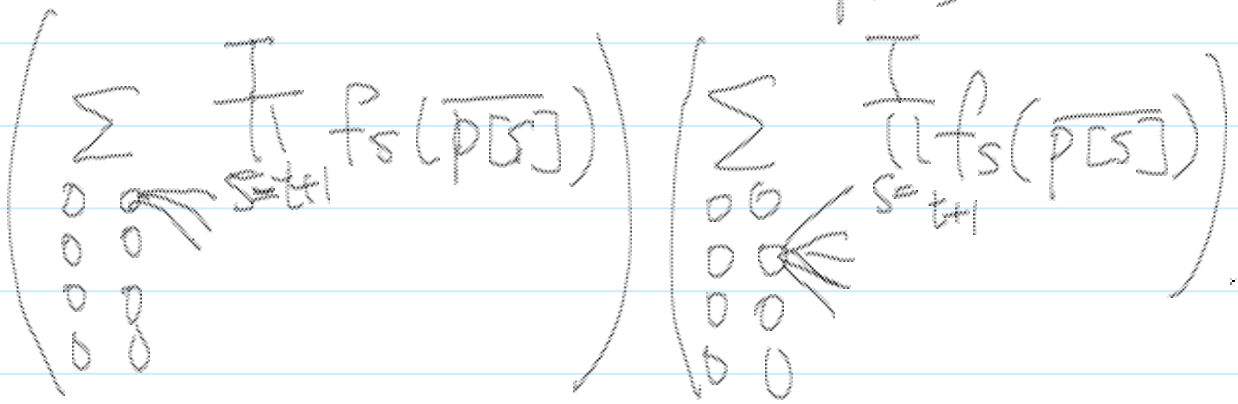
The BCJR algorithm 1974

* Again, we rely on iterative computation.
* Iteration must be performed in two ways.
** The forward iteration computes

$$\left(\sum_{\cdots} \prod_{t=1}^{\square} f_t(p[\square]) \right) \left(\sum_{\cdots} \prod_{t=1}^{\square} \cdot f_s(p[\square]) \right)$$



** The backward iteration computes



Detailed description of the BCJR algorithm

① Given the observation $\vec{y}_{obs}$
Compute the metric $m_{s,s'}$ for each path segment from state $s \rightarrow$ state s'. (The same step as in the VA.)

② The forward iteration. Compute the forward state metric

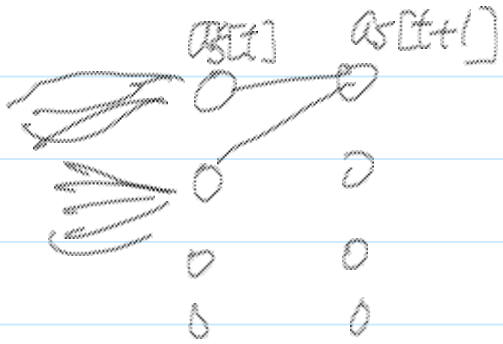
$\alpha_s[t]$

Initialization

$\alpha_{00}[0] = 1$, $\alpha_s[0] = 0$ for all $s \neq 00$

$\forall t = 0, \ldots, T-1$,

$$\alpha_s[t+1] = \sum_{s' \text{ that reaches } s} \alpha_{s'}[t] \cdot m_{s',s}$$



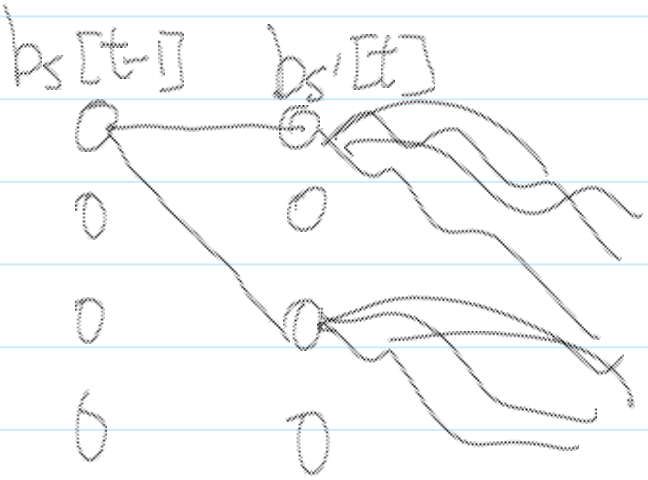
③ Backward iteration. Compute $b_s[t]$

Initialization

$b_s[T] = 1$ for all s

$\forall t = T, \ldots, 1$.

$$b_s[t-1] = \sum_{s' \text{ that is reachable from } s} b_{s'}[t]\, m_{s,s'}$$



④ Make the decision for the i-th bit by $a_s[t]\, b_s[t-1]$.

$$\sum_{p_i = x} \prod_{t=1} f_t(p[t])$$

$$= \sum_{\text{path } s \to s' \text{ s.t. } i_{t_0} = x} a_s[t_0]\, m_{s,s'}\, b_{s'}[t_0+1]$$

where the i-th bit is in the t_0-th path segment.

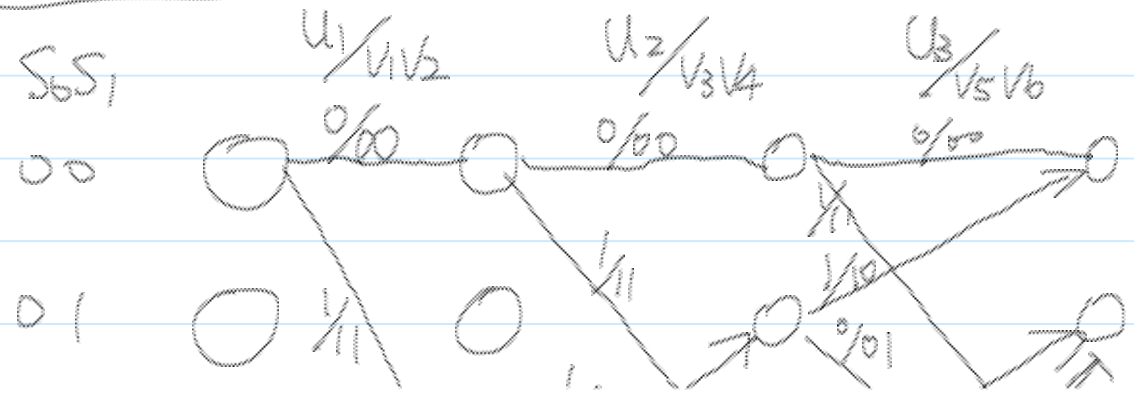
Then we can find

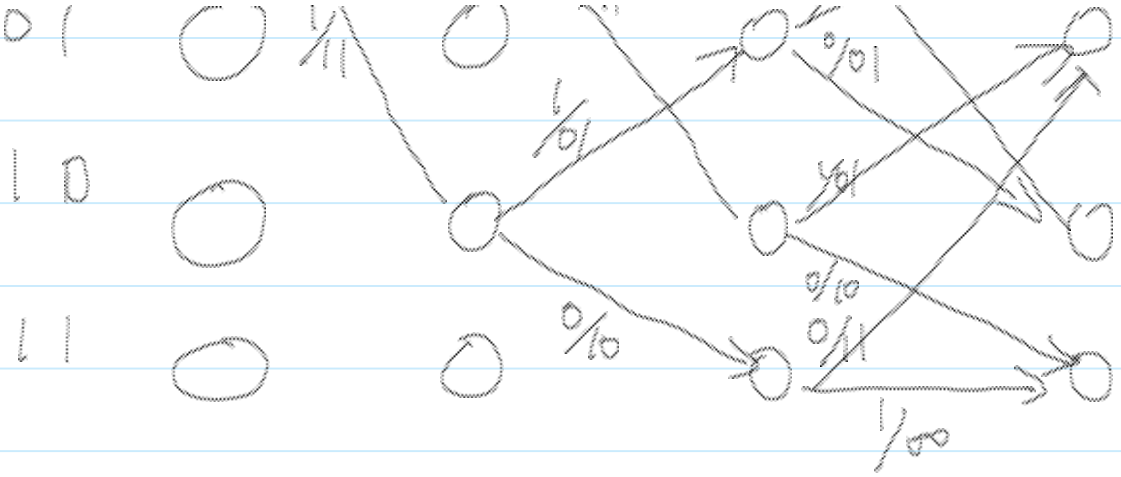
$$\underset{x=0,1}{\operatorname{argmax}} \sum_{P_i=x} \prod_{t=1}^{I} f_t(p_{[t]})$$

* BCJR finds the optimal bit-level ML decoder.
* The BCJR algorithm can be easily modified to be a ML decoder for the info bit u & for the coded bits, v. by carefully counting the paths in the final comparison stage.

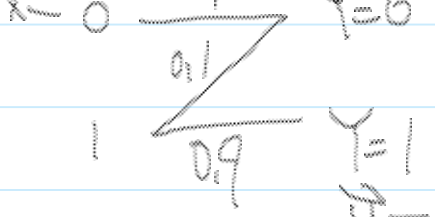
The principle of the BCJR decoder can be extended to take into account of different prior distributions.

An illustrating example With the addition info of the prior distri P_U





Channel: Z-channel



$\vec{y}$ = the input into BKS

The observation is 00 10 01

but suppose we know that (u_1, u_2, u_3) are not uniform Bernoulli distributed but have $P(U=1)=0.05$ $P(U=0)=0.95$

Q: Use the BCJR algorithm to find the MAP u_2 value.

Ans: We are interested in maximizing

$$\arg\max_{\cdot} \sum_{\cdot} P_{\vec{Y}} \cdot U(\vec{y}, \vec{x})$$

$$\underset{u_2=0 \text{ or } 1}{\operatorname{argmax}} \sum_{\text{all } \vec{u} \text{ w. the given } u_2 \text{ value}} \underbrace{\Pr(\vec{y}, \vec{u})}_{\text{the joint prob.}}$$

$$= \underset{u_2=0 \text{ or } 1}{\operatorname{argmax}} \sum_{\substack{\text{all paths} \\ \text{with the} \\ \text{given } u_2 \\ \text{value}}} \left(\prod_{t=1}^{3} P_{Y_{2t-1}Y_{2t}|V_{2t-1}V_{2t}} \right) \cdot P_{u_1}P_{u_2}P_{u_3}$$

$$= \underset{u_2=0 \text{ or } 1}{\operatorname{argmax}} \sum_{\substack{\text{all paths} \\ \text{with the} \\ \text{given } u_2 \\ \text{value}}} \left(\prod_{t=1}^{3} \left(P_{Y_{2t-1}Y_{2t}|V_{2t-1}V_{2t}} \cdot P_{u_t} \right) \right)$$

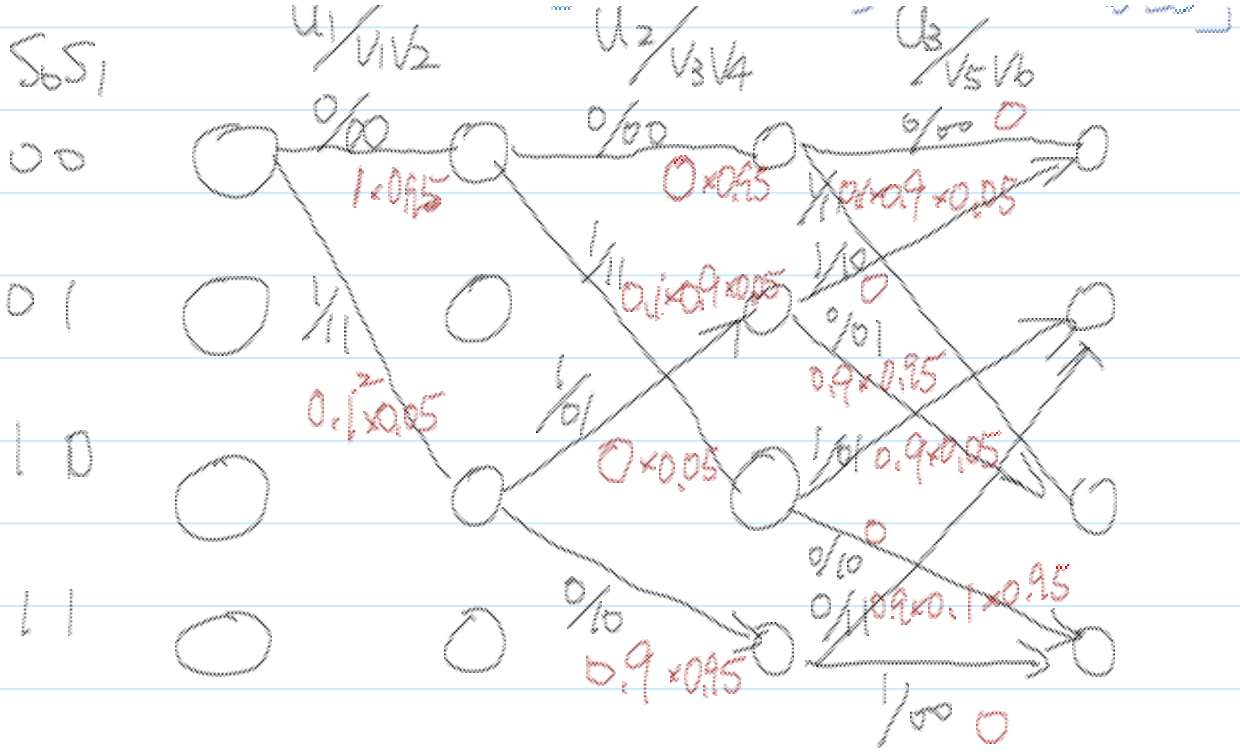
$\Longrightarrow$ use $f_t(\cdot)$ instead

$$= \underset{u_2=0 \text{ or } 1}{\operatorname{argmax}} \sum_{\text{all paths using } u_2} \prod_{t=1}^{3} f_t(\overline{p[t]})$$

Observation: 00 10 01

$a_5[0]$ $a_5[1]$ $a_5[2]$ $a_5[3]$

S_0S_1 u_1/v_1v_2 u_2/v_3v_4 u_3/v_5v_6



Step 1: Initialize the path metrics

Step 2: Compute the forward metrics.

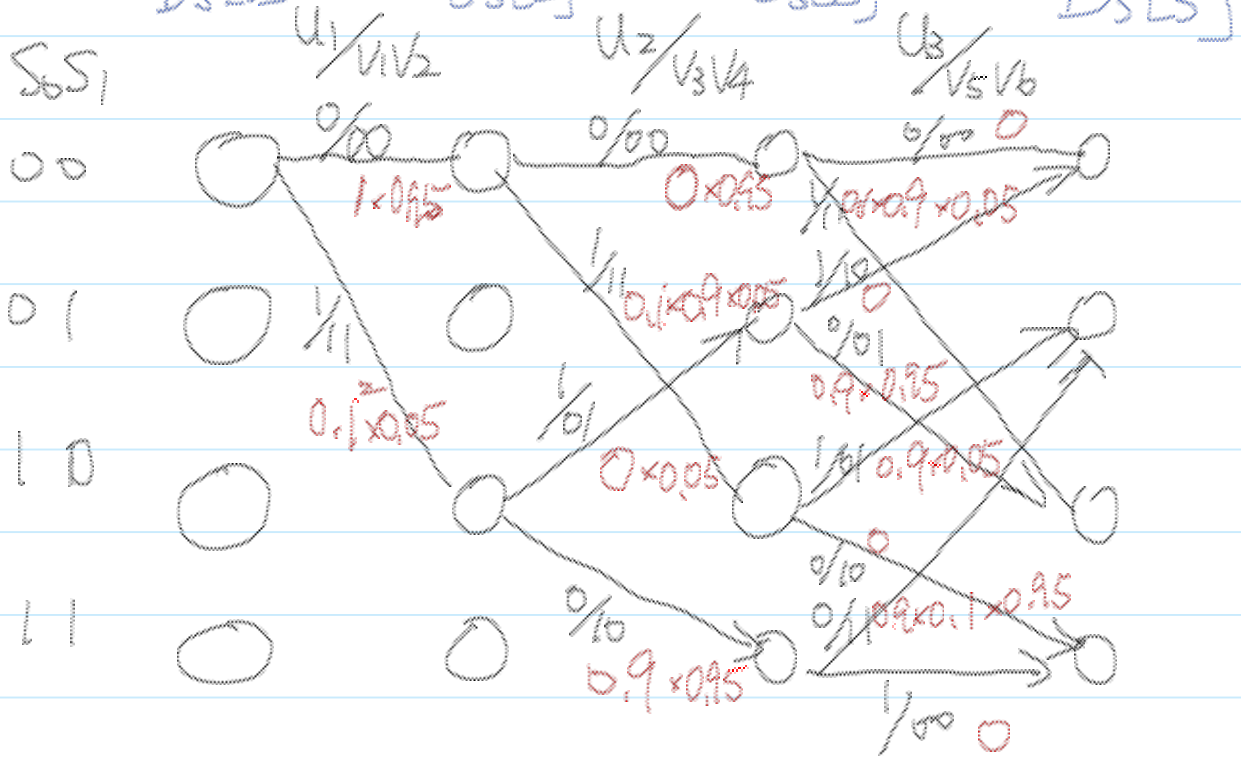
$$\alpha_s[t+1] = \sum_{s':s' \to s} \alpha_{s'}[t] \cdot m_{s',s}$$

$\alpha_s[t]$

1	0.95	0	0
0	0	0	5.5178715×10^{-4}
0	5×10^{-4}	4.275×10^{-3}	0
0	0	4.275×10^{-4}	0

Observation: 00 10 01

$b_s[0]$ $b_s[1]$ $b_s[2]$ $b_s[3]$



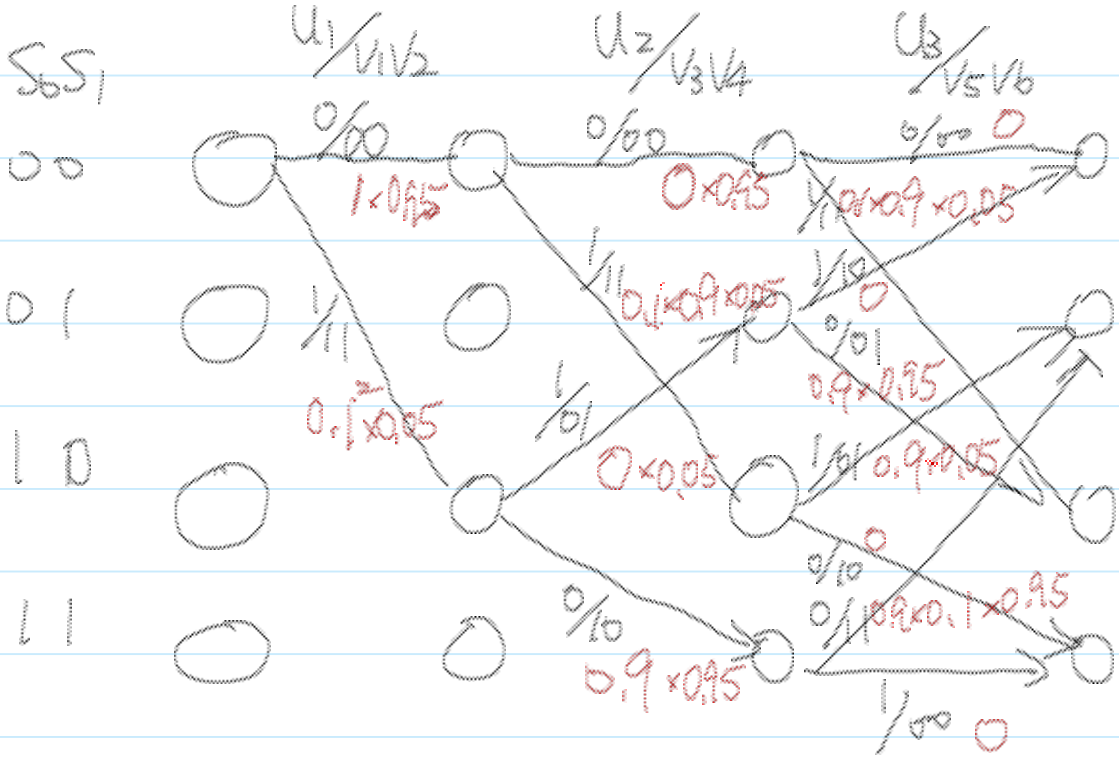
Step 3: Compute the backward metrics.

$$b_s[t-1] = \sum b_{s'}[t]\, m_{s,s'}$$

$b_s[t]$			$b_s[3]$
2.2781625×10^{-4}	2.025×10^{-4}	4.5×10^{-3}	1
0	0	8.55×10^{-4}	1
0	7.3125×10^{-2}	4.5×10^{-2}	1
0			

0 0 8.55×10^{-2} 1

Observation: 00 10 01



Step 4: Make a decision on U_2

Two cases: $U_2=0$ $U_2=1$

$\alpha_{00}[1] \cdot m_{00,00} b_{00}[2]$

$+ \alpha_{10}[1] \cdot m_{10,11} b_{11}[2]$

$= 3.65125 \times 10^{-5}$

$\alpha_{00}[1] \cdot m_{00,10} b_{10}[2]$

$+ \alpha_{10}[1] \cdot m_{10,01} b_{01}[2]$

$= 1.9375 \times 10^{-4}$

$\Rightarrow$ The MAP for U_2 is 1.

With high posterior prob.

$$\frac{1.92375 \times 10^{-9}}{3.65125 \times 10^{-5} + 1.92375 \times 10^{-9}} \doteq 84\%$$

* Although both being linear with respect to T, the complexity of BCJR $\geq$ VA (needs a lot of multiplication & summation)
* In practice, BCJR is used less frequently as the bit error rate improvement over the VA does not justify the complexity.
* Recently, BCJR is getting more attention due to the invention of the "turbo codes."
* In practice, we simply "approximate" the iteration of the VA & the BCJR. Common simplification techniques: quantized as[t] & bs[t] values

Common simplification techniques.

② quantized $as[t]$ & $bs[t]$ values

③ Incomplete sum / max / product. The best k rule.

④ Proper rescaling. $as'[t] \leftarrow const \cdot as[t]$

$bs'[t] \leftarrow const \cdot bs[t]$

since our decision, based on the relative values of $as[t]$ ($bs[t]$) for different s value rescaling does not change the decision

⑤ Implementation by table look-up.