Hidden Markov Models

Model: $\mu = \langle A, B, \Pi \rangle$, over a fixed set of states, S, and output alphabet, K.
There are three fundamental questions we can ask:

1. **Output Probability**: Given μ, what is the probability of seeing $O = o_1 \ldots o_T$?

   $$P(O \mid \mu)$$

2. **State Sequence Decoding**: Given O and μ, what is the most probable state sequence, X, that produced it?

   $\text{argmax}_{x_1 \ldots x_T} P(x_1 \ldots x_T \mid O, \mu)$
   (not $\max_{x_t} P(x_t \mid O, \mu)$ for all $1 \leq t \leq T$!)

3. **Parameter Estimation**: Given O_{train}, and a range of possible μ, M, which $\mu \in M$ is most likely to have produced O_{train}?

   $$\underset{\mu}{\text{argmax}}\, P(O_{\text{train}} \mid \mu)$$

Output Probability

$$
\begin{aligned}
P(O \mid \mu) &= \sum_X P(O \mid X, \mu) P(X \mid \mu) \\
&= \sum_{x_1 \dots x_T} \pi_{x_1} b_{x_1 o_1} \prod_{t=2}^{T} a_{x_{t-1} x_t} b_{x_t o_t}
\end{aligned}
$$

Naive algorithm: $\mathcal{O}(T \cdot N^T)$ multiplications,
$\mathcal{O}(N^T)$ additions

Bad move

We can instead use dynamic programming

Output Probability: Forward Algorithm

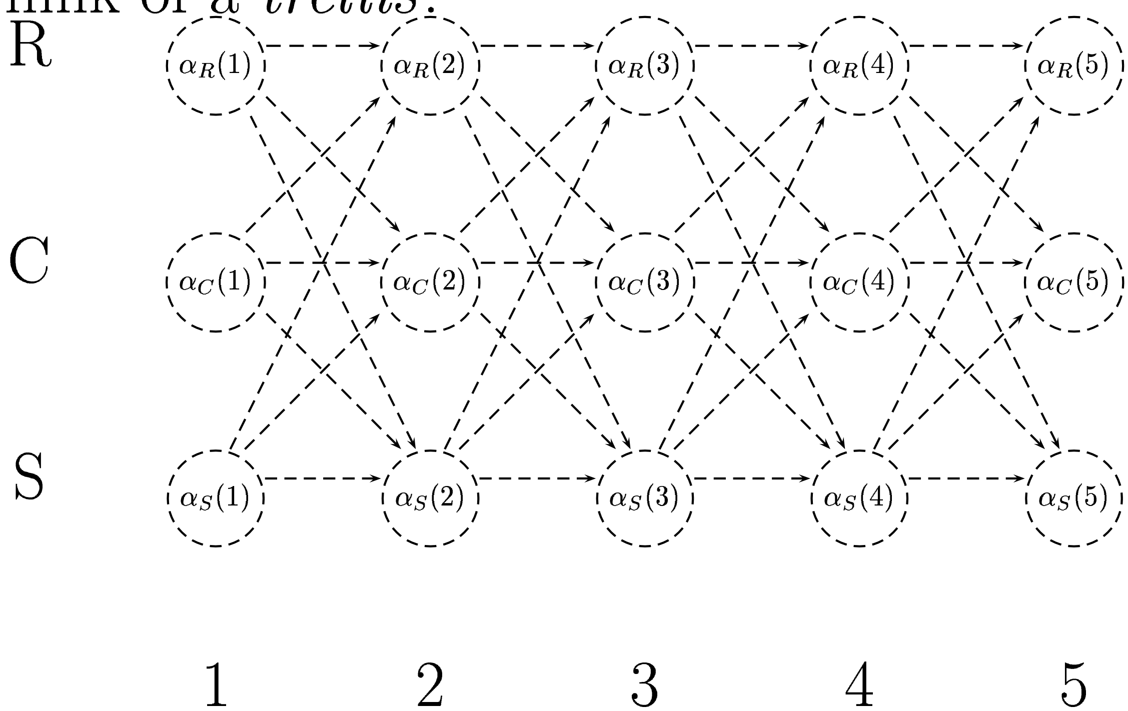
Let $\alpha_i(t) = P(o_1 \dots o_t, x_t = s_i \mid \mu)$. Then:

$$\alpha_i(1) = \pi_i b_{io_1}$$
$$\alpha_i(t+1) = \sum_{j=1}^{N} \alpha_j(t) a_{ji} b_{io_{t+1}}$$

$$P(O \mid \mu) = \sum_{i=1}^{N} \alpha_i(T)$$

$\mathcal{O}(T \cdot N^2)$ multiplications, $\mathcal{O}(TN)$ additions.

Think of a *trellis*:



Output Probability: Backward Algorithm

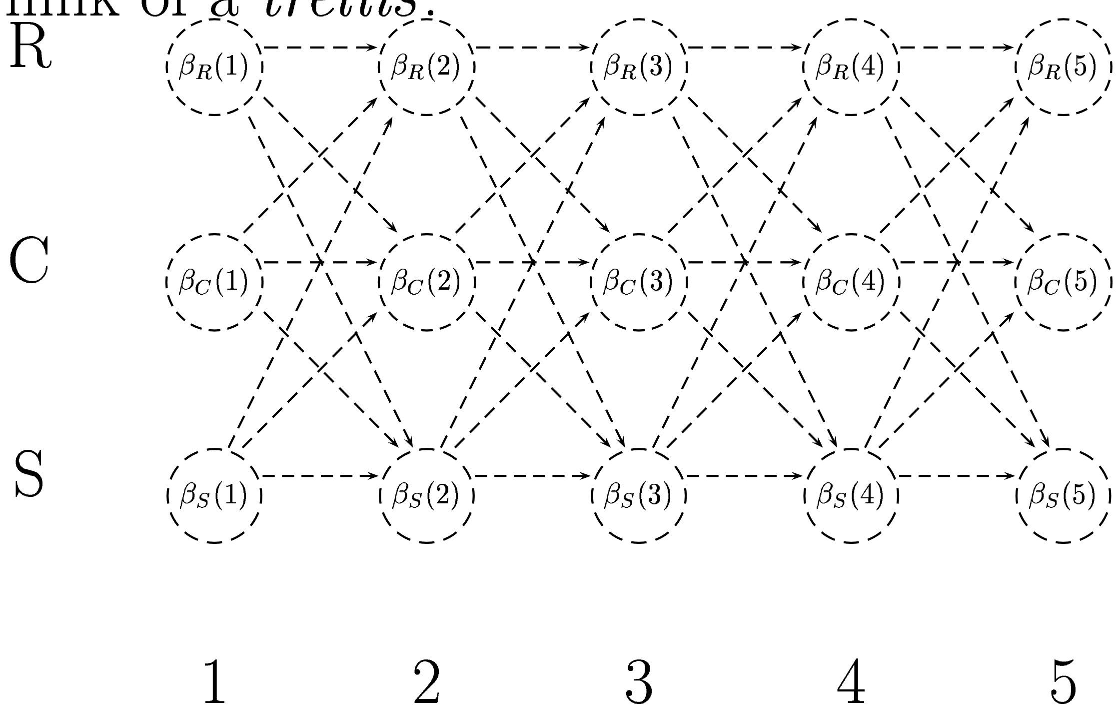
Let $\beta_i(t) = P(o_{t+1} \dots o_T \mid x_t = s_i, \mu)$. Then:

$$\beta_i(T) = 1$$

$$\beta_i(t) = \sum_{j=1}^{N} b_{jo_{t+1}} a_{ij} \beta_j(t+1)$$

$$P(O \mid \mu) = \sum_{i=1}^{N} \pi_i b_{io_1} \beta_i(1)$$

Think of a *trellis*:

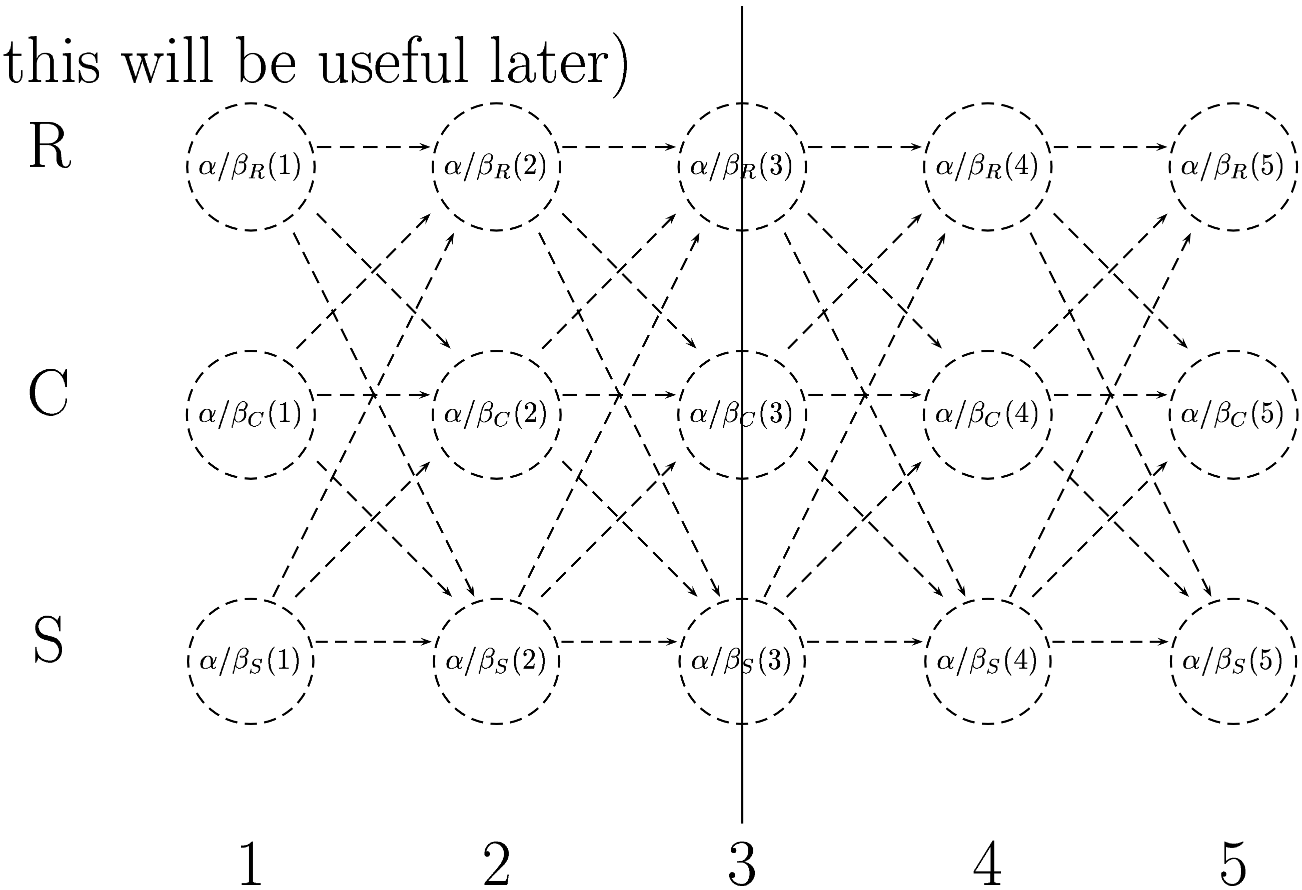


Output Probability

In fact, for any t:

$$P(O \mid \mu) = \sum_{i=1}^{N} \alpha_i(t)\beta_i(t)$$

(this will be useful later)



State Sequence Decoding: Viterbi Algorithm

$$\operatorname*{argmax}_{x_1 \dots x_T} P(x_1 \dots x_T \mid O, \mu)$$

Same idea: use dynamic programming

Let $\delta_i(t) = \max_{x_1 \dots x_{t-1}} P(x_1 \dots x_{t-1}, o_1 \dots o_t, x_t = s_i \mid \mu)$
and $\psi_i(t)$ the previous state in this optimal path.
Then:

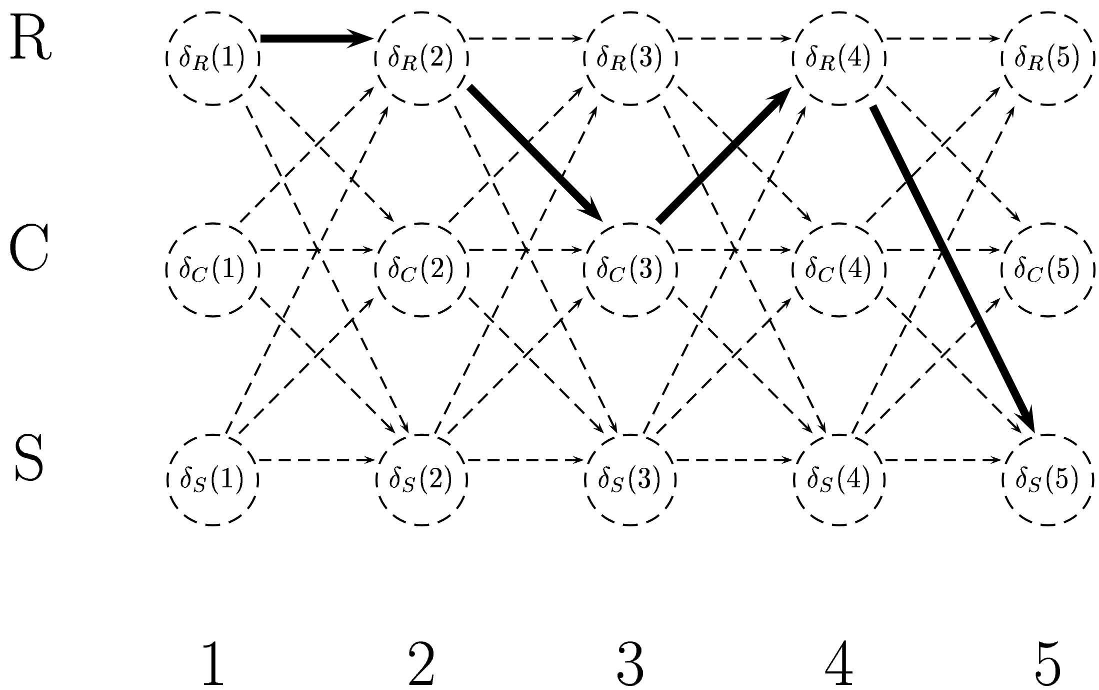
$$\delta_i(1) = \pi_i b_{io_1}$$
$$\delta_i(t+1) = \max_{1 \le j \le N} \delta_j(t) a_{ji} b_{io_{t+1}}$$
$$\psi_i(t+1) = \operatorname*{argmax}_{1 \le j \le N} \delta_j(t) a_{ji} b_{io_{t+1}}$$

Then work backwards to find optimal state sequence:

$$P(\hat{x}_1 \dots \hat{x}_T \mid O, \mu) = \delta_{\hat{x}_T}(T)$$
$$\hat{x}_T = \operatorname*{argmax}_{1 \le i \le N} \delta_i(T)$$
$$\hat{x}_t = \psi_{\hat{x}_{t+1}}(t+1)$$

State Sequence Decoding: Viterbi Algorithm

Let $\delta_i(t) = \max_{x_1 \ldots x_{t-1}} P(x_1 \ldots x_{t-1}, o_1 \ldots o_t, x_t = s_i \mid \mu)$
and $\psi_i(t)$ the previous state in this optimal path.



Parameter Re-estimation: Baum-Welch Algorithm

$$\operatorname*{argmax}_{\mu} P(O_{\text{train}} \mid \mu)$$

Let $p_t(i,j)$ be probability of traversing from state i to state j at time $t+1$, given output O, and $\gamma_i(t)$ be probability of traversing through state i at time t, given output O.
Then we should set:

$$\begin{aligned}
\hat{\pi}_i &= \text{rel freq of } s_i \text{ at } t = 1 \\
&= \gamma_i(1) \\
\hat{a}_{ij} &= \frac{\text{rel freq of transitions } s_i \text{ to } s_j}{\text{rel freq of transitions from } s_i} \\
&= \frac{\sum_{t=1}^{T-1} p_t(i,j)}{\sum_{t=1}^{T-1} \gamma_i(t)} \\
\hat{b}_{ik} &= \frac{\text{rel freq of transitions from } s_i \text{ with output } k}{\text{rel freq of transitions from } s_i} \\
&= \frac{\sum_{\substack{t=1 \\ o_t = k}}^{T-1} \gamma_i(t)}{\sum_{t=1}^{T-1} \gamma_i(t)}
\end{aligned}$$

Parameter Re-estimation: Baum-Welch Algorithm

How do we calculate $p_t(i,j)$ and $\gamma_i(t)$?
Tabulate $\xi_t(i,j) = \alpha_i(t) \cdot a_{ij} \cdot b_{jo_{t+1}} \cdot \beta_j(t+1)$, for all i, j, t. Then:

$$
\begin{aligned}
p_t(i,j) &= \frac{\alpha_i(t) \cdot a_{ij} \cdot b_{jo_{t+1}} \cdot \beta_j(t+1)}{P(O|\mu)} \\
&= \frac{\alpha_i(t) \cdot a_{ij} \cdot b_{jo_{t+1}} \cdot \beta_j(t+1)}{\sum_{m=1}^{N} \alpha_m(t)\ \beta_m(t)} \\
&= \frac{\alpha_i(t) \cdot a_{ij} \cdot b_{jo_{t+1}} \cdot \beta_j(t+1)}{\sum_{m=1}^{N} \sum_{n=1}^{N} \alpha_m(t) \cdot a_{mn} \cdot b_{no_{t+1}} \cdot \beta_n(t+1)} \\
&= \frac{\xi_t(i,j)}{\sum_{m=1}^{N} \sum_{n=1}^{N} \xi_t(m,n)} \\
\gamma_i(t) &= \sum_{j=1}^{N} p_t(i,j)
\end{aligned}
$$

Parameter Re-estimation: Baum-Welch Algorithm

$$\hat{\mu} = \langle \hat{\pi}, \hat{A}, \hat{B} \rangle$$

- We are guaranteed that $P(O \mid \hat{\mu}) \geq P(O \mid \mu)$
- Iterate until convergence — but we might get stuck in local maximum or saddle point.
- In practice, Baum-Welch is very sensitive to the initial model, particularly the choice of B.