AIL 7022: Reinforcement Learning

Lecture 2: Hidden Markov Models

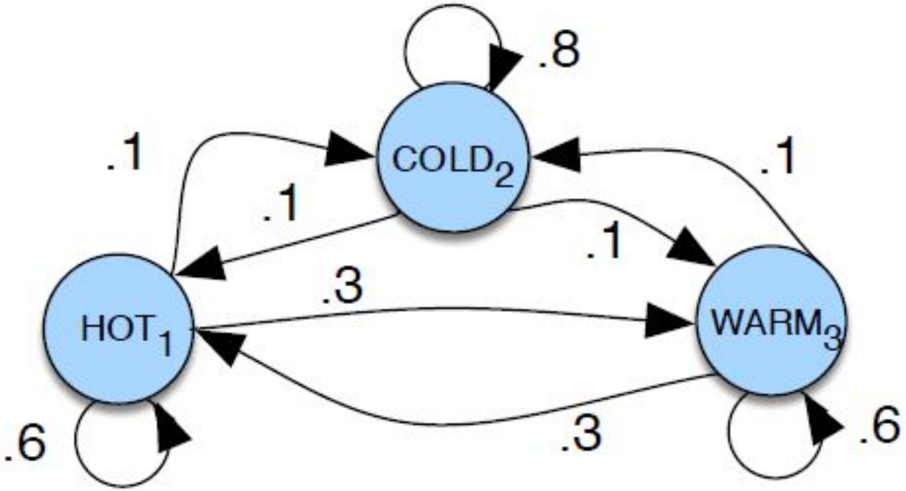
Instructor: Raunak Bhattacharyya

YARDI SCHOOL OF ARTIFICIAL INTELLIGENCE
INDIAN INSTITUTE OF TECHNOLOGY DELHI

Why HMMs?

- Uncertainty: Zone of probabilistic reasoning
- Foundational material towards MDPs
- Constructs: Sequences of states, a.k.a. trajectories
- Algorithms: Iterative approaches
- Using observed data to make inferences

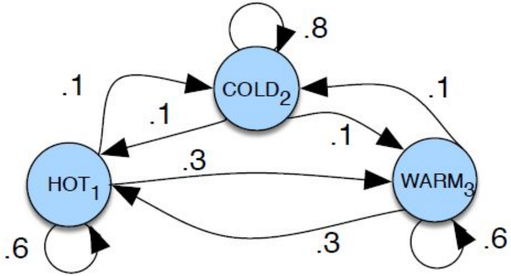
Markov Chain



Weather

Source: [SLP, Dan Jurafsky]

Markov Chain



$\mathcal{S} = \{s_1, s_2, \ldots, s_N\}$ A set of N states

$$T = \begin{pmatrix} p_{11} & p_{12} & \cdots & p_{1N} \\ p_{21} & p_{22} & \cdots & p_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ p_{N1} & p_{N2} & \cdots & p_{NN} \end{pmatrix}$$ A transition probability matrix

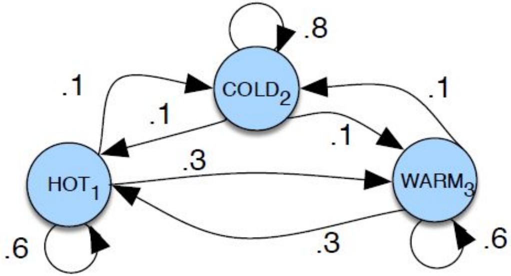
$\rho = \{p(s_1), p(s_2), \ldots, p(s_N)\}$ Initial state distribution

Markov Chain

$$p(s_i = a \mid s_1, s_2, \ldots, s_{i-1}) = p(s_i = a \mid s_{i-1})$$

Markov Property

Exercise



➔ Compute the probability of the sequences
- Cold, Cold, Cold, Cold
- Cold, Hot, Cold, Hot

What information is missing from this question?

Initial State Distribution

Hidden Markov Model

$$\mathcal{S} = \{s_1, s_2, \ldots, s_N\}$$

$$T = \begin{pmatrix} p_{11} & p_{12} & \cdots & p_{1N} \\ p_{21} & p_{22} & \cdots & p_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ p_{N1} & p_{N2} & \cdots & p_{NN} \end{pmatrix}$$

$$\rho = \{p(s_1), p(s_2), \ldots, p(s_N)\}$$

$\mathcal{O} = \{o_1, o_2, \ldots, o_M\}$ A set of M possible observations

$B = \begin{pmatrix} b_{11} & b_{12} & \cdots & b_{1M} \\ b_{21} & b_{22} & \cdots & b_{2M} \\ \vdots & \vdots & \ddots & \vdots \\ b_{N1} & b_{N2} & \cdots & b_{NM} \end{pmatrix}$ Observation probability matrix, where $b_{ij} = p(o_j \mid s_i)$

$p(o_i \mid s_1, \ldots, s_i, \ldots, s_T, o_1, \ldots, o_i, \ldots, o_T) = p(o_i \mid s_i)$ **Output Independence**

Hidden Markov Model

$$O = \{o_1, o_2, \ldots, o_T\}$$

Input to the HMM: A sequence of T observations

$$\mathcal{S} = \{s_1, s_2, \ldots, s_N\}$$

$$T = \begin{pmatrix} p_{11} & p_{12} & \cdots & p_{1N} \\ p_{21} & p_{22} & \cdots & p_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ p_{N1} & p_{N2} & \cdots & p_{NN} \end{pmatrix}$$

$$\rho = \{p(s_1), p(s_2), \ldots, p(s_N)\}$$

$$\mathcal{O} = \{o_1, o_2, \ldots, o_M\}$$

$$B = \begin{pmatrix} b_{11} & b_{12} & \cdots & b_{1M} \\ b_{21} & b_{22} & \cdots & b_{2M} \\ \vdots & \vdots & \ddots & \vdots \\ b_{N1} & b_{N2} & \cdots & b_{NM} \end{pmatrix}$$

Where are HMMs used?

• Speech Recognition	Acoustic Signal	Phenomes: Pat/Bat
• Activity Recognition	Sensor Readings	Activity: walking
• Music Transcription	Audio features	Musical notes
• Finance	Financial indicators	Bull, Bear, Stable

HMM: Three Fundamental Problems

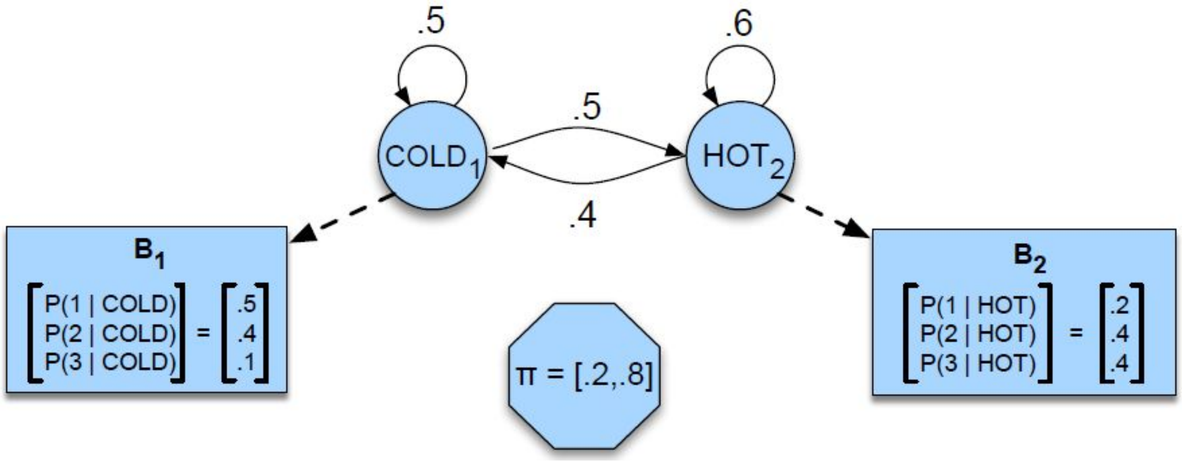
Problem 1 (Likelihood):
Given an HMM $\lambda = (\mathcal{T}, \mathcal{B})$ and an observation sequence O, determine the likelihood $P(O \mid \lambda)$.

Problem 2 (Decoding):
Given an observation sequence O and an HMM $\lambda = (\mathcal{T}, \mathcal{B})$, discover the best hidden state sequence.

Problem 3 (Learning):
Given an observation sequence O and the set of states in the HMM, learn the HMM parameters $\mathcal{T}$ and $\mathcal{B}$.

Running Example: Weather and Ice Cream

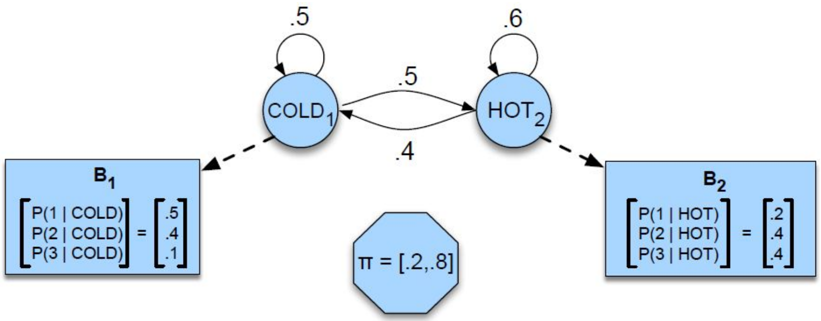
➔ Given a sequence of observations O (each an integer representing the number of ice creams eaten on a given day) find the 'hidden' sequence of weather states (H or C)



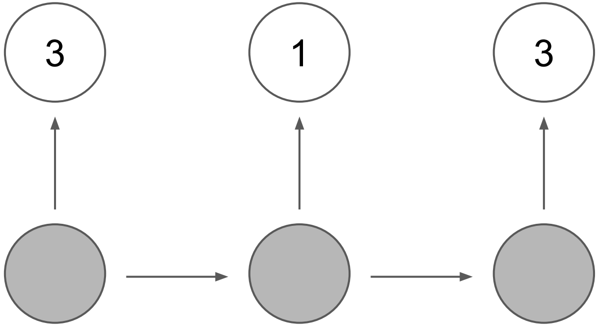
Example from SLP, Dan Jurafsky

Problem 1: Likelihood Computation

Likelihood Computation

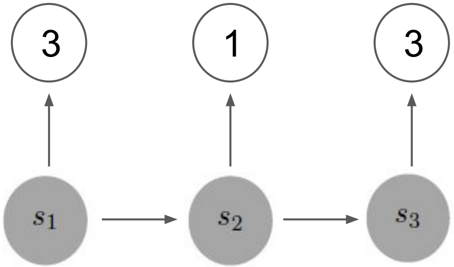


- Given the ice-cream eating HMM, what is the probability of the sequence of ice creams eaten being 3, 1, 3?
  - 3 ice creams on day 1, 1 on day 2, and 3 on day 3



$$p(3, 1, 3)?$$

Marginalise over Hidden State Seq.



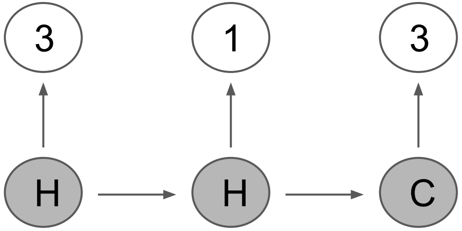
$$p(O) = p(o_1, o_2, \ldots, o_T) \qquad p(S) = p(s_1, s_2, \ldots, s_T)$$

$$p(O) = \sum_S p(O, S)$$

$$= \sum_S p(O \mid S) \cdot p(S)$$

$$p(O \mid S) = \prod_{i=1}^{T} p(o_i \mid s_i)$$

Known Hidden State Seq.



$$p(O \mid S) = \prod_{i=1}^{T} p(o_i \mid s_i)$$

$$p(3, 1, 3 \mid H, H, C)$$

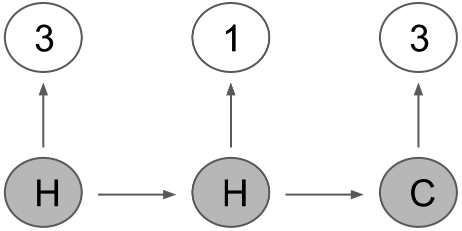
$$= p(o_1 = 3, o_2 = 1, o_3 = 3 \mid s_1 = H, s_2 = H, s_3 = C)$$

$$= p(o_1 = 3 \mid s_1 = H, s_2 = H, s_3 = C) \cdot p(o_2 = 1, o_3 = 3 \mid o_1 = 3, s_1 = H, s_2 = H, s_3 = C)$$

$$= p(o_1 = 3 \mid s_1 = H) \; p(o_2 = 1, o_3 = 3 \mid o_1 = 3, s_1 = H, s_2 = H, s_3 = C)$$

$$p(3, 1, 3 \mid H, H, C) = p(3 \mid H) \cdot p(1 \mid H) \cdot p(3 \mid C)$$

Marginalise via Enumeration



$$p(O) = \sum_{S} p(O, S)$$

$$p(3,1,3) = p(3,1,3,C,C,C) + p(3,1,3,C,C,H) + p(3,1,3,C,H,H) + \dots$$

$$= p(3,1,3 \mid C,C,C) \cdot p(C,C,C) + p(3,1,3 \mid C,C,H) \cdot p(C,C,H) + \dots$$

$$p(3,1,3 \mid C,C,C) = p(3 \mid C) \cdot p(1 \mid C) \cdot p(3 \mid C)$$

$$p(C,C,C) = p(s_i = C) \cdot p(C \mid C) \cdot p(C \mid C)$$

Brute Force Enumeration

Algorithm 1 Brute Force Likelihood

1: Enumerate all possible hidden state sequences (N^T of them)
2: **for** each hidden state sequence **do**
3: Compute $p(o_{1:T} \mid s_{1:T})p(s_{1:T})$
4: **end for**
5: Add up all the above obtained $p(o_{1:T}, s_{1:T})$ to marginalize over all possible hidden state sequences

Do you see a problem with this approach?

Forward Algorithm

$$\alpha_t(j) = p(o_1, o_2, \ldots, o_t, s_t = j)$$

1. Initialization:

$$\alpha_1(j) = p(s_1 = j) \cdot p(o_1 \mid s_1 = j) \qquad 1 \leq j \leq N$$

2. Recursion:

$$\alpha_t(j) = \sum_{i=1}^{N} \alpha_{t-1}(i) \cdot p(s_t = j \mid s_{t-1} = i) \cdot p(o_t \mid s_t = j) \qquad 1 \leq j \leq N, 1 < t \leq T$$

3. Termination:

$$p(O) = \sum_{i=1}^{N} \alpha_T(i) \qquad \sum_{i=1}^{N} p(o_1, \ldots, o_T, s_T = i)$$

Marginalise over final state

Forward Algo: Rationale

$$\alpha_t(j) = p(o_1, o_2, \ldots, o_t, s_t = j)$$

$$= \sum_{i=1}^{N} p(o_1, o_2, \ldots, o_{t-1}, s_{t-1} = i, o_t, s_t = j)$$

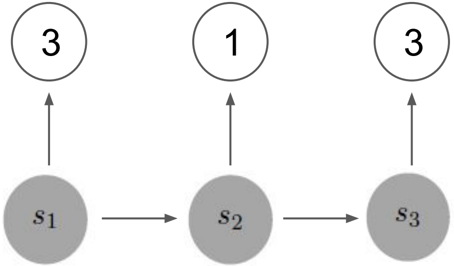
$$= \sum_{i=1}^{N} p(o_{1:t-1}, s_{t-1} = i) \cdot p(o_t, s_t = j \mid o_{1:t-1}, s_{t-1} = i)$$

$$= \sum_{i=1}^{N} p(o_{1:t-1}, s_{t-1} = i) \cdot p(s_t = j \mid o_{1:t-1}, s_{t-1} = i) \cdot p(o_t \mid o_{1:t-1}, s_{t-1} = i, s_t = j)$$

$$= \sum_{i=1}^{N} p(o_{1:t-1}, s_{t-1} = i) \cdot p(o_t \mid s_t = j) \cdot p(s_t = j \mid s_{t-1} = i)$$

$$\alpha_t(j) = \sum_{i=1}^{N} \alpha_{t-1}(i) \cdot p(s_t = j \mid s_{t-1} = i) \cdot p(o_t \mid s_t = j)$$

Brute Force vs. Forward Algorithm



$$p(o_{1:T}) = \sum_{\forall \{s_{1:T}\}} p(o_{1:T}, s_{1:T})$$

$$= \sum_{\forall \{s_{1:T}\}} p(o_{1:T} | s_{1:T}) \cdot p(s_{1:T})$$

$$= \sum_{\forall \{s_{1:T}\}} \left(\prod_{i=1}^{T} p(o_i \mid s_i) \cdot p(s_{1:T}) \right)$$

$$p(o_{1:T}) = \sum_{i=1}^{N} p(o_{1:T}, s_T = s^i)$$

$$= \sum_{i=1}^{N} p(o_{1:T}, s_T = i)$$

Notation alert

Forward Algorithm

$$\alpha_t(j) = p(o_1, o_2, \ldots, o_t, s_t = j)$$

1. Initialization:

$$\alpha_1(j) = p(s_1 = j) \cdot p(o_1 \mid s_1 = j) \qquad 1 \le j \le N$$

2. Recursion:

$$\alpha_t(j) = \sum_{i=1}^{N} \alpha_{t-1}(i) \cdot p(s_t = j \mid s_{t-1} = i) \cdot p(o_t \mid s_t = j) \qquad 1 \le j \le N, 1 < t \le T$$

3. Termination:

$$p(O) = \sum_{i=1}^{N} \alpha_T(i)$$

How do we translate this to pseudocode?

Forward Algo: Pseudocode

$$\alpha_t(j) = p(o_1, o_2, \ldots, o_t, s_t = j)$$

function FORWARD(*observations* of len T, *state-graph* of len N) **returns** *forward-prob*

create a probability matrix *forward[N,T]*
for each state s **from** 1 **to** N **do**
  $forward[s,1] \leftarrow \pi_s * b_s(o_1)$
for each time step t **from** 2 **to** T **do**
  for each state s **from** 1 **to** N **do**

$$forward[s,t] \leftarrow \sum_{s'=1}^{N} forward[s',t-1] * a_{s',s} * b_s(o_t)$$

$$forwardprob \leftarrow \sum_{s=1}^{N} forward[s,T]$$

return *forwardprob*

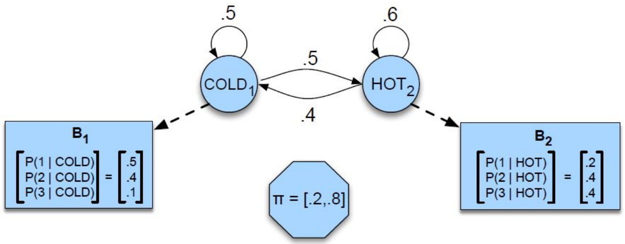
$$\alpha_1(j) = p(s_1 = j) \cdot p(o_1 \mid s_1 = j)$$

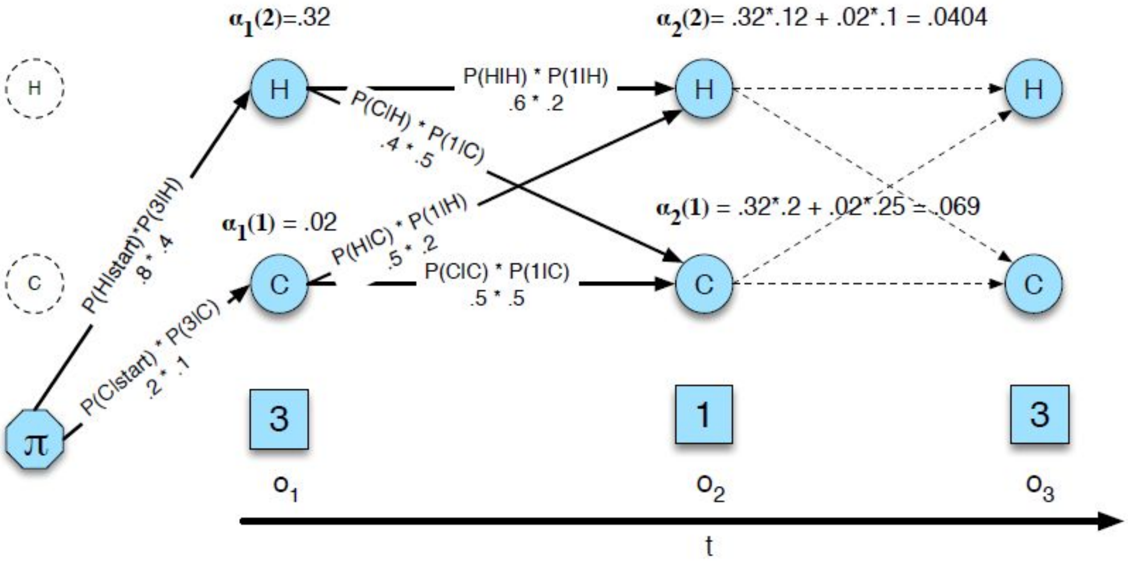
$$\alpha_t(j) = \sum_{i=1}^{N} \alpha_{t-1}(i) \cdot p(s_t = j \mid s_{t-1} = i) \cdot p(o_t \mid s_t = j)$$

$$p(O) = \sum_{i=1}^{N} \alpha_T(i)$$

Source: [SLP, Dan Jurafsky]

Forward Trellis





How do you compute the original goal: observation sequence likelihood?

Source: SLP, Dan Jurafsky

HMM: Three Fundamental Problems

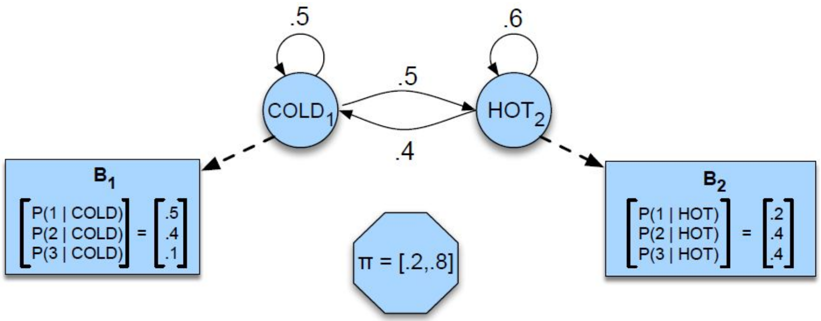
Problem 1 (Likelihood):
Given an HMM $\lambda = (\mathcal{T}, \mathcal{B})$ and an observation sequence O, determine the likelihood $P(O \mid \lambda)$.

Problem 2 (Decoding):
Given an observation sequence O and an HMM $\lambda = (\mathcal{T}, \mathcal{B})$, discover the best hidden state sequence.

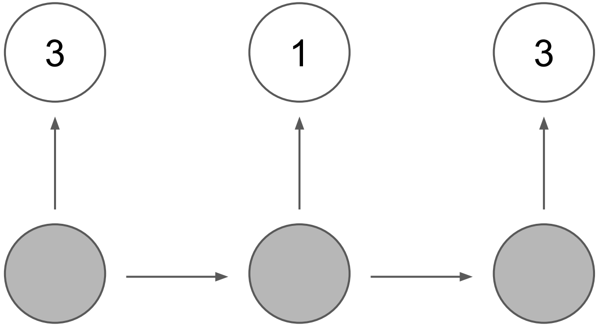
Problem 3 (Learning):
Given an observation sequence O and the set of states in the HMM, learn the HMM parameters $\mathcal{T}$ and $\mathcal{B}$.

Problem 2: Decoding

Problem 2: Decoding



- ➔ Find the hidden state sequence that was most likely to have generated the input observation sequence.



$p(3, 1, 3)?$ **Earlier goal**

What should the goal be now?

$$\arg\max_{s_1, s_2, s_3} p(s_1, s_2, s_3 \mid o_1 = 3, o_2 = 1, o_3 = 3)?$$

Decoding: Equivalent Objective

$$\arg\max_{s_1,s_2,s_3} p(s_1, s_2, s_3 \mid o_1 = 3, o_2 = 1, o_3 = 3)?$$

$$p(s_1, s_2, s_3 \mid o_1, o_2, o_3) = \frac{p(o_1, o_2, o_3, s_1, s_2, s_3)}{p(o_1, o_2, o_3)}$$

$$\arg\max_{s_1,s_2,s_3} p(s_1, s_2, s_3 \mid o_1, o_2, o_3) = \arg\max_{s_1,s_2,s_3} p(o_1, o_2, o_3, s_1, s_2, s_3)$$

Where have we seen this quantity before?

$$p(o_{1:3}, s_{1:3})$$

$$p(o_{1:3}) = \sum_{\forall\{s_{1:3}\}} p(o_{1:3}, s_{1:3})$$

Can you think of a brute force algorithm for decoding?

Viterbi Algorithm: Building Blocks

$$\arg\max_{s_1,s_2,s_3} p(o_1,o_2,o_3,s_1,s_2,s_3)$$

$$\text{Define: } v_t(j) = \max_{s_1,s_2,\ldots,s_{t-1}} p(s_1,s_2,\ldots,s_{t-1},o_1,o_2,\ldots,o_t,s_t=j)$$

$$v_t(j) = \max_{s_{1:t-1}} p(s_{1:t-1},o_{1:t},s_t=j)$$

How do we use this to get what we want?

$$\text{Goal: } \max_{s_1,s_2,s_3} p(s_1,s_2,s_3,o_1,o_2,o_3)$$

$$\max_{s_1,s_2} p(s_1,s_2,o_1,o_2,o_3,s_3=C) \qquad \max_{s_1,s_2} p(s_1,s_2,o_1,o_2,o_3,s_3=H)$$

$$\max_{i=1}^{N} \max_{s_1,s_2} p(s_1,s_2,o_1,o_2,o_3,s_3=i)$$

$$\text{Goal obtained by: } \max_{i=1}^{N} v_T(i)$$

Viterbi Algo: Recursion

$$v_t(j) = \max_{s_{1:t-1}} p(s_{1:t-1}, o_{1:t}, s_t = j)$$

$$p(s_{1:t-1}, o_{1:t}, s_t = j) = p(s_{1:t-2}, o_{1:t-1}, s_{t-1}, o_t, s_t = j)$$

$$= p(s_{1:t-2}, o_{1:t-1}, s_{t-1}) \cdot p(o_t, s_t = j \mid s_{1:t-2}, o_{1:t-1}, s_{t-1})$$

$$= p(s_{1:t-2}, o_{1:t-1}, s_{t-1}) \cdot p(s_t = j \mid s_{1:t-2}, o_{1:t-1}, s_{t-1}) \cdot p(o_t \mid s_t = j, s_{1:t-2}, o_{1:t-1}, s_{t-1})$$

$$= p(s_{1:t-2}, o_{1:t-1}, s_{t-1}) \cdot p(s_t = j \mid s_{t-1}) \cdot p(o_t \mid s_t = j)$$

$$p(s_{1:t-1}, o_{1:t}, s_t = j) = p(s_{1:t-2}, o_{1:t-1}, s_{t-1}) \cdot p(s_t = j \mid s_{t-1}) \cdot p(o_t \mid s_t = j)$$

Viterbi Algo: Recursion

$$p(s_{1:t-1}, o_{1:t}, s_t = j) = p(s_{1:t-2}, o_{1:t-1}, s_{t-1}) \cdot p(s_t = j \mid s_{t-1}) \cdot p(o_t \mid s_t = j)$$

$$\max_{s_{1:t-1}} p(s_{1:t-1}, o_{1:t}, s_t = j)$$

$$= \max_{i=1}^{N} \max_{s_{1:t-2}} p(s_{1:t-2}, o_{1:t-1}, s_{t-1} = i) \cdot p(s_t = j \mid s_{t-1} = i) \cdot p(o_t \mid s_t = j)$$

$$v_t(j) = \max_{i=1}^{N} v_{t-1}(i) \cdot t_{ij} \cdot b_j(o_t)$$

Viterbi Algorithm

1. **Initialization:**

$$
\begin{aligned}
v_1(j) &= \pi_j b_j(o_1) \qquad 1 \le j \le N \\
bt_1(j) &= 0 \qquad\qquad\quad\; 1 \le j \le N
\end{aligned}
$$

2. **Recursion**

$$
\begin{aligned}
v_t(j) &= \max_{i=1}^{N} v_{t-1}(i)\, a_{ij}\, b_j(o_t); \quad 1 \le j \le N, 1 < t \le T \\
bt_t(j) &= \operatorname*{argmax}_{i=1}^{N} v_{t-1}(i)\, a_{ij}\, b_j(o_t); \quad 1 \le j \le N, 1 < t \le T
\end{aligned}
$$

3. **Termination:**

$$
\begin{aligned}
\text{The best score:} \quad P* &= \max_{i=1}^{N} v_T(i) \\
\text{The start of backtrace:} \quad q_T* &= \operatorname*{argmax}_{i=1}^{N} v_T(i)
\end{aligned}
$$

Source: SLP, Dan Jurafsky

Viterbi Algorithm: Pseudocode

create a path probability matrix *viterbi[N,T]*
for each state *s* **from** 1 **to** *N* **do**
  $viterbi[s,1] \leftarrow \pi_s * b_s(o_1)$
  $backpointer[s,1] \leftarrow 0$

$$v_1(j) = \pi_j b_j(o_1) \qquad 1 \le j \le N$$
$$bt_1(j) = 0 \qquad 1 \le j \le N$$

for each time step *t* **from** 2 **to** *T* **do** ; recursion step
  for each state *s* **from** 1 **to** *N* **do**
    $viterbi[s,t] \leftarrow \max_{s'=1}^{N} viterbi[s',t-1] * a_{s',s} * b_s(o_t)$
    $backpointer[s,t] \leftarrow \operatorname{argmax}_{s'=1}^{N} viterbi[s',t-1] * a_{s',s} * b_s(o_t)$

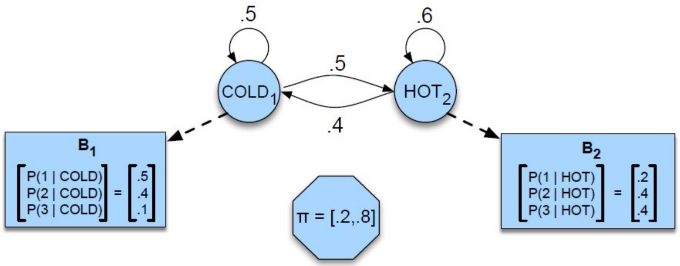
$$v_t(j) = \max_{i=1}^{N} v_{t-1}(i)\, a_{ij}\, b_j(o_t);$$
$$bt_t(j) = \operatorname{argmax}_{i=1}^{N} v_{t-1}(i)\, a_{ij}\, b_j(o_t)$$

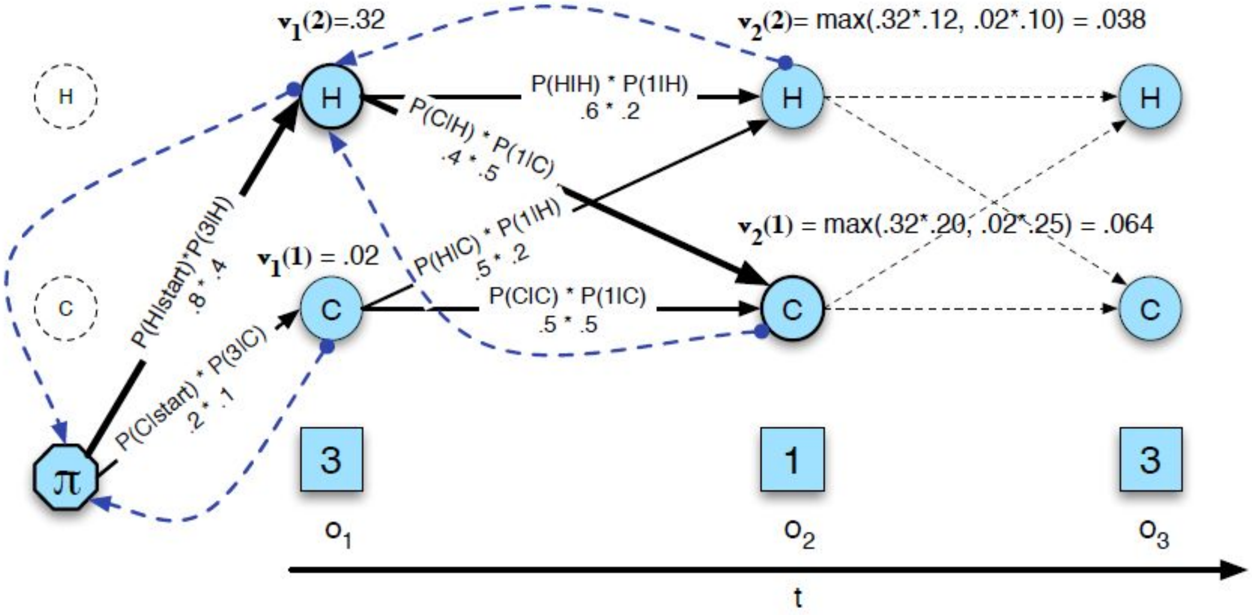
$bestpathprob \leftarrow \max_{s=1}^{N} viterbi[s,T]$; termination step
$bestpathpointer \leftarrow \operatorname{argmax}_{s=1}^{N} viterbi[s,T]$; termination step
$bestpath \leftarrow$ the path starting at state *bestpathpointer*, that follows backpointer[] to states back in time

$$P* = \max_{i=1}^{N} v_T(i)$$
$$q_T* = \operatorname{argmax}_{i=1}^{N} v_T(i)$$

Source: SLP, Dan Jurafsky

Viterbi Trellis





Source: SLP, Dan Jurafsky