AIL 722: Reinforcement Learning

Lecture 5: Baum-Welch

Raunak Bhattacharyya

YARDI SCHOOL OF ARTIFICIAL INTELLIGENCE
INDIAN INSTITUTE OF TECHNOLOGY DELHI

Outline

- Consolidation: Observation Prob. and Transition Prob.
- Baum-Welch Algorithm
- Introducing Decisions into the Markov Model: MDPs
- Example MDPs

Consolidation

$$\alpha_t(i) = p(o_{1:t}, s_t = i)$$

$$\beta_t(i) = p(o_{t+1:T} \mid s_t = i)$$

Observation

$$\hat{p}(o^k \mid s^j) = \frac{\text{expected number of times in state } j \text{ and observing } o^k}{\text{expected number of times in state } j}$$

$$\gamma_t(j) = p(s_t = j \mid o_{1:T})$$

Expected state occupancy count

$$p(o_{1:T}, s_t = j) = \alpha_t(j) \cdot \beta_t(j)$$

Transition

$$\hat{p}(s^j \mid s^i) = \frac{\text{expected number of transitions from state } i \text{ to state } j}{\text{expected number of transitions from state } i}$$

$$\xi_t(i,j) = p(s_t = i, s_{t+1} = j \mid o_{1:T})$$

Expected state transition count

$$p(s_t = i, s_{t+1} = j, o_{1:T}) = \alpha_t(i) \cdot a_{ij} \cdot b_j(o_{t+1}) \cdot \beta_{t+1}(j)$$

Does this require knowing transition and observation?

Can't we just use alpha?

Consolidation

$$\alpha_t(i) = p(o_{1:t}, s_t = i)$$

$$\beta_t(i) = p(o_{t+1:T} \mid s_t = i)$$

Observation

Transition

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

$$\gamma_t(j) = \frac{\alpha_t(j).\beta_t(j)}{\sum_{i=1}^{N} \alpha_T(i)}$$

$$\xi_t(i,j) = \frac{\alpha_t(i) \cdot a_{ij} \cdot b_j(o_{t+1}) \cdot \beta_{t+1}(j)}{\sum_{i=1}^{N} \alpha_T(i)}$$

$$\hat{p}(o^k \mid s^j) = \frac{\sum_{t=1}^{T} \gamma_t(j) 1(o_t = o_k)}{\sum_{t=1}^{T} \gamma_t(j)}$$

$$\hat{p}(s^j \mid s^i) = \frac{\sum_{t=1}^{T-1} \xi_t(i,j)}{\sum_{t=1}^{T-1} \sum_{k=1}^{N} \xi_t(i,k)}$$

Baum-Welch Algorithm

Baum-Welch Algorithm

function FORWARD-BACKWARD(*observations* of len T, *output vocabulary* V, *hidden state set* Q) **returns** *HMM=(A,B)*

initialize A and B
iterate until convergence
E-step

$$\gamma_t(j) = \frac{\alpha_t(j)\beta_t(j)}{\alpha_T(q_F)} \quad \forall\, t \text{ and } j$$

$$\xi_t(i,j) = \frac{\alpha_t(i)\, a_{ij} b_j(o_{t+1})\beta_{t+1}(j)}{\alpha_T(q_F)} \quad \forall\, t,\ i, \text{ and } j$$

M-step

$$\hat{a}_{ij} = \frac{\sum_{t=1}^{T-1} \xi_t(i,j)}{\sum_{t=1}^{T-1} \sum_{k=1}^{N} \xi_t(i,k)}$$

$$\hat{b}_j(v_k) = \frac{\sum_{t=1 s.t.\ O_t=v_k}^{T} \gamma_t(j)}{\sum_{t=1}^{T} \gamma_t(j)}$$

return A, B

Alert: Use of q for state

Expected state occupancy and state transition counts

Re-estimate the transition and observation probabilities

Source: SLP, Dan Jurafsky

Finding Forward and Backward Probs

Algorithm 3 Baum-Welch Algorithm

Input:
Observation sequence: $O = (O_1, O_2, \ldots, O_T)$
Number of states: N
Number of distinct observation symbols: M
Initial model parameters: transition probabilities A, emission probabilities B, and initial state probabilities π
Output: Updated model parameters: A, B, π
Initialize parameters A, B, π
repeat
  Forward Procedure:
  Initialize α:
  for $i = 1$ to N **do**

$$\alpha_1(i) = \pi_i b_i(O_1)$$

  end for
  Recursively compute α:
  for $t = 2$ to T **do**
    for $j = 1$ to N **do**

$$\alpha_t(j) = \left(\sum_{i=1}^{N} \alpha_{t-1}(i) a_{ij} \right) b_j(O_t)$$

    end for
  end for

Initial State Distribution?

  Backward Procedure:
  Initialize β:
  for $i = 1$ to N **do**

$$\beta_T(i) = 1$$

  end for
  Recursively compute β:
  for $t = T - 1$ to 1 **do**
    for $i = 1$ to N **do**

$$\beta_t(i) = \sum_{j=1}^{N} a_{ij} b_j(O_{t+1}) \beta_{t+1}(j)$$

    end for
  end for

E-Step and M-step: Gamma and Obs Prob

```
Compute γ:
for t = 1 to T do
    for i = 1 to N do
```

$$\gamma_t(i) = \frac{\alpha_t(i)\beta_t(i)}{\sum_{j=1}^{N} \alpha_t(j)\beta_t(j)}$$

```
    end for
end for
```

```
Re-estimate B:
for j = 1 to N do
    for k = 1 to M do
```

$$b_j(k) = \frac{\sum_{t=1}^{T} \delta(O_t = k)\gamma_t(j)}{\sum_{t=1}^{T} \gamma_t(j)}$$

where $\delta(\cdot)$ is 1 if the argument is true, 0 otherwise

```
    end for
end for
```

Baum-Welch Properties

- Each iteration changes the parameters in a way that is guaranteed to increase the likelihood of the data

- Anytime algorithm: Can stop at any time prior to convergence to get an approximate solution

- Converges to a local maximum

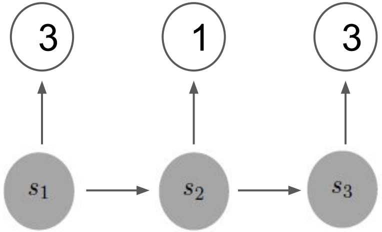
HMM: Three Fundamental Problems

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 $\mathcal{B}$.

Review



Likelihood

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

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

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

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

Decoding

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

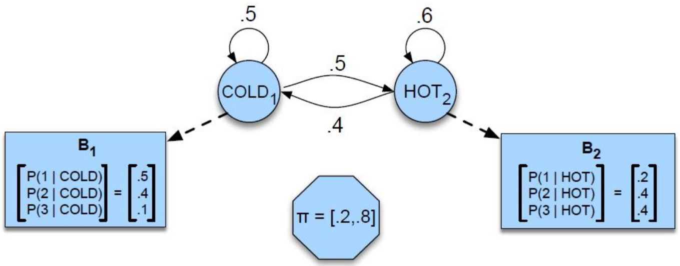
$$\arg\max_{s_1, s_2, s_3} p(o_1, o_2, o_3, s_1, s_2, s_3)$$

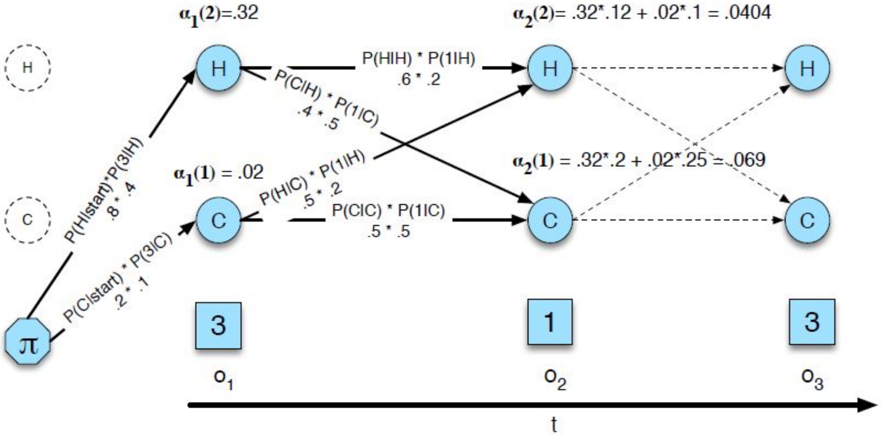
$$v_t(j) = \max_{s_{1:t-1}} p(s_{1:t-1}, o_{1:t}, s_t = j)$$

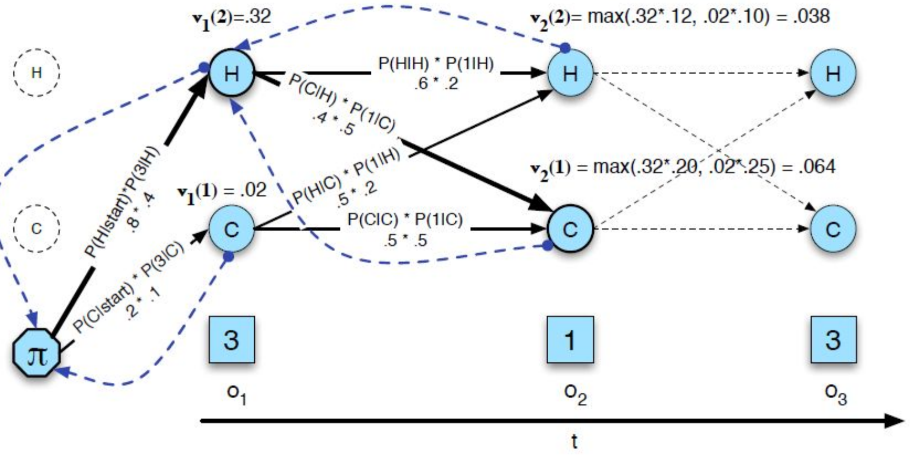
$$v_t(j) = \max_{i=1}^{N} v_{t-1}(i) \cdot t_{ij} \cdot b_j(o_t)$$

$$\max_{i=1}^{N} v_T(i)$$

Trellis: Forward vs. Viterbi







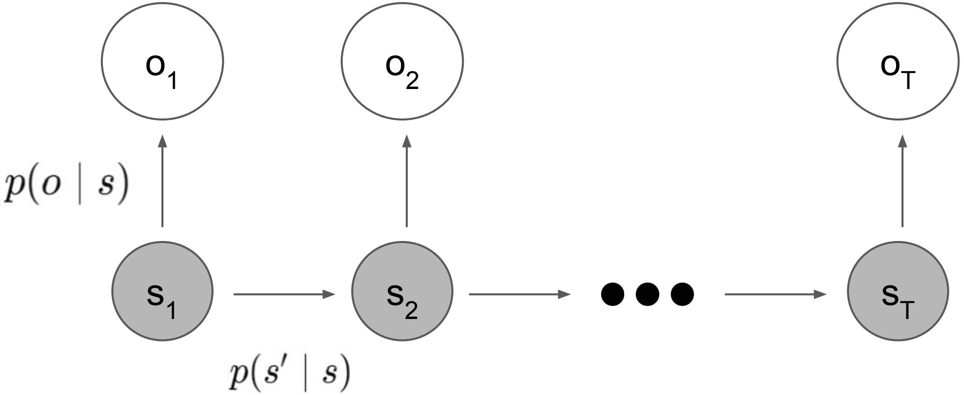
Source: SLP, Dan Jurafsky

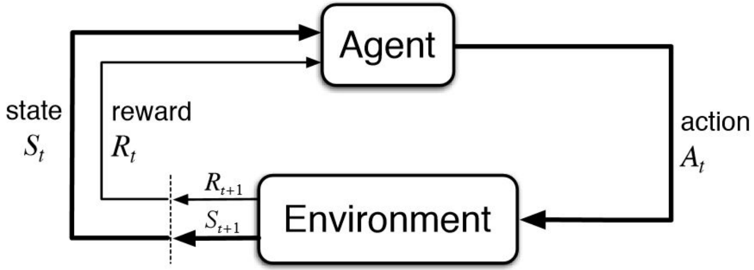
Why HMMs?

- Uncertainty: Zone of probabilistic reasoning
- Constructs: Sequences of states, a.k.a. trajectories
- Algorithms: Dynamic Programming, Expectation Maximisation
- States, Transitions and Observations

Markov Decision Processes

HMM: State Evolution





Source: [Sutton & Barto]

MDP

$$s_t, a_t, r(s_t, a_t)$$

$$\text{MDP : Tuple}\langle \mathcal{S}, \mathcal{A}, T, R, \rho \rangle$$

$\mathcal{S}$: State Space

$\mathcal{A}$: Action Space

Difference from HMM?

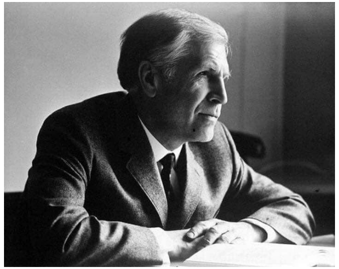
$T : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$: Probabilistic Transition Function

$R : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$: Reward Function

ρ : Initial State Distribution

Richard Bellman, Source: Wikipedia

$$x_t, u_t, c(x_t, u_t)$$



Lev Pontryagin, Source: Wikipedia

MDP: State Evolution