AIL 7022: Reinforcement Learning

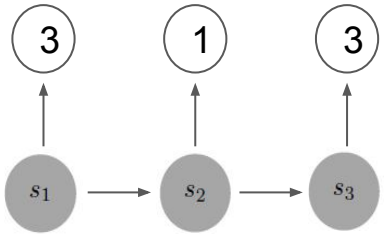
Lecture 3: Stochastic Sequential Models (Part 2)

Instructor: Raunak Bhattacharyya

Review

Likelihood

Decoding



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

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

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

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

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

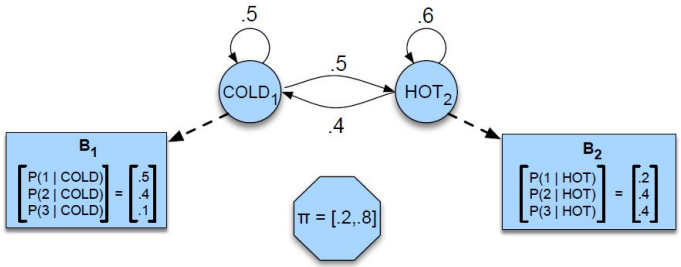
$$\alpha_t(j) = \sum_{i=1}^{N} \alpha_{t-1}(i) \cdot p(s_t = j \mid s_{t-1} = i) \cdot p(o_t \mid s_t = j)$$

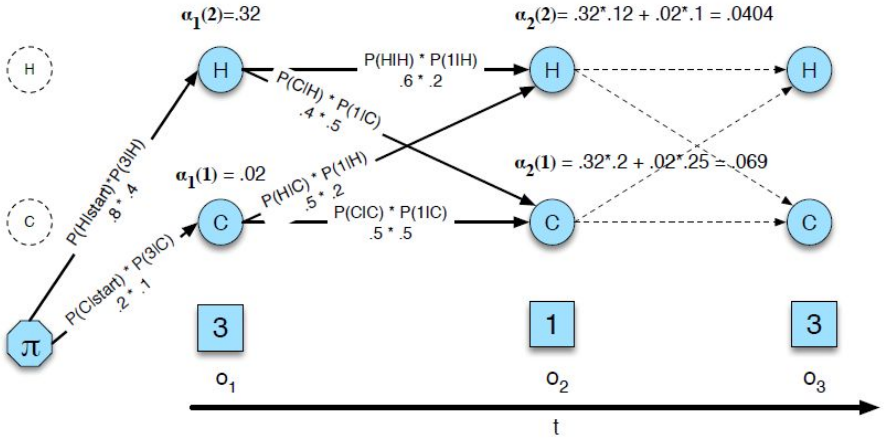
$$v_t(j) = \max_{i=1}^{N} v_{t-1}(i) \cdot t_{ij} \cdot b_j(o_t)$$

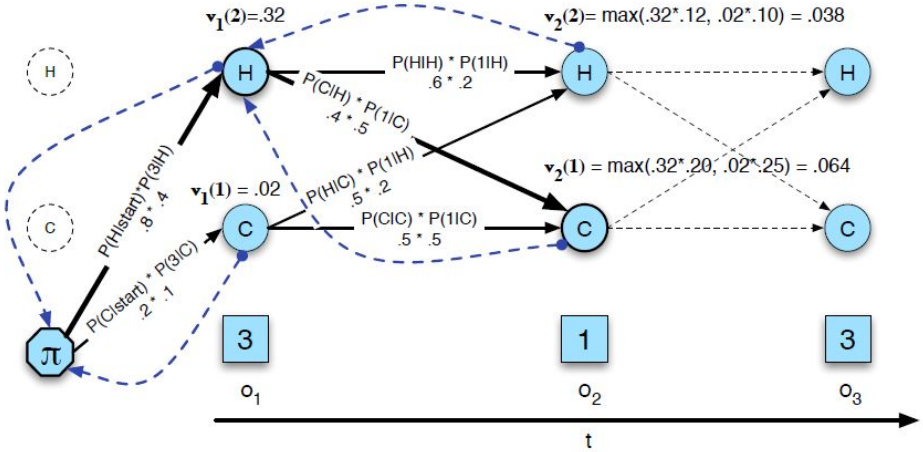
$$p(O) = \sum_{i=1}^{N} \alpha_T(i)$$

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

Trellis: Forward vs. Viterbi







DP = recursion + memoization?

Source: SLP, Dan Jurafsky

Plan

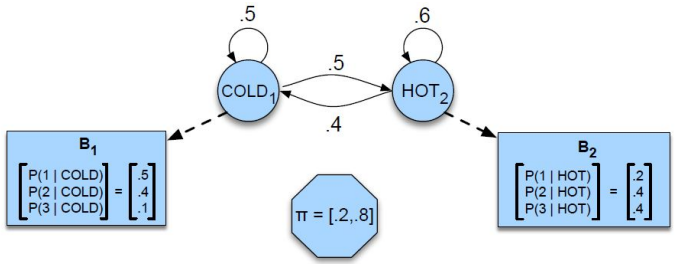
- HMM learning
- Intro to MDPs
- MDP formulations

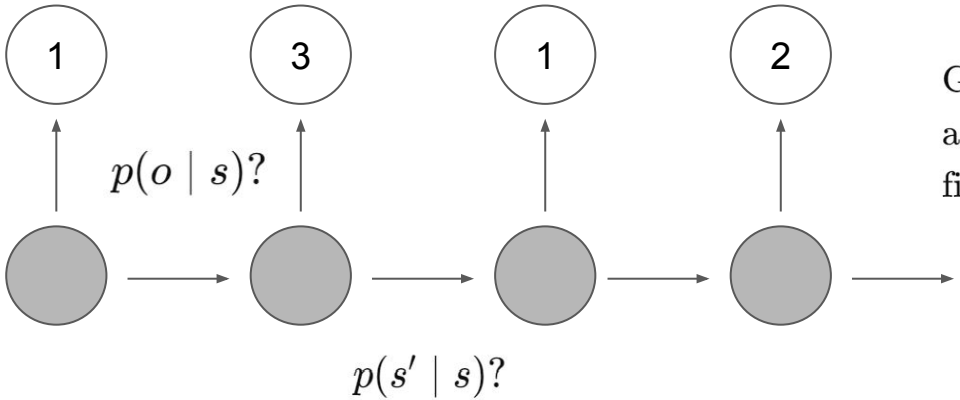
HMM Learning

Outline

- Observation Probability
- Backward Probability
- Transition Probability
- Baum-Welch Algorithm

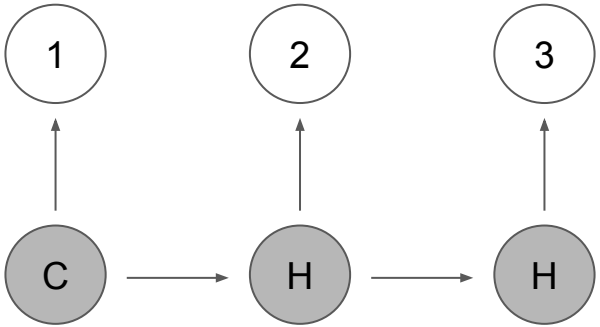
Problem 3: Learning

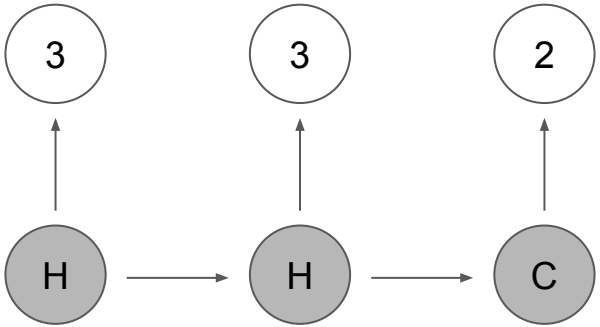


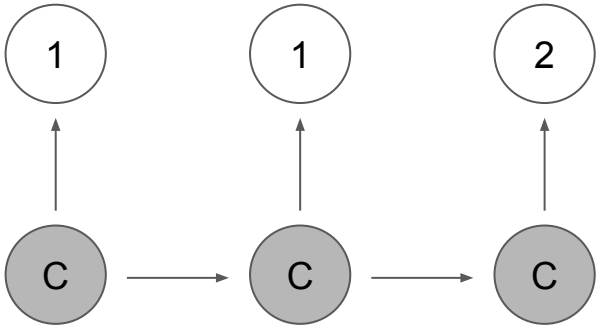


Given $O = \{1, 3, 1, 2, 3, 1, \ldots\}$,
and the set of possible hidden states $\{H, C\}$,
find the T and B matrices.

Simple Case: Fully Visible Markov Model







$$p(s_1 = H) = \frac{1}{3}$$

$$p(s_1 = C) = \frac{2}{3}$$

$$p(H \mid H) = \frac{2}{3} \qquad p(H \mid C) = \frac{1}{3}$$

$$p(C \mid H) = \frac{1}{3} \qquad p(C \mid C) = \frac{2}{3}$$

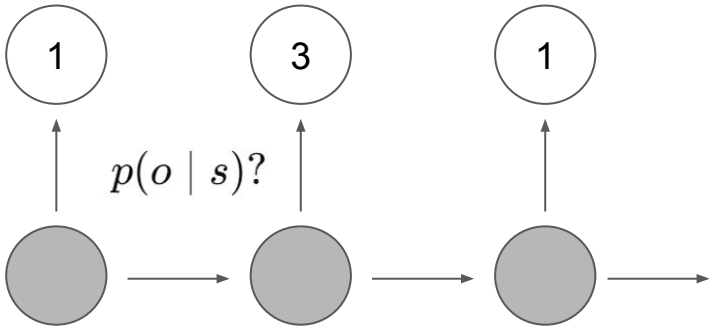
$$p(1 \mid H) = 0 \qquad p(1 \mid C) = 0.6$$
$$p(2 \mid H) = 0.25 \qquad p(2 \mid C) = 0.4$$
$$p(3 \mid H) = 0.75 \qquad p(3 \mid C) = 0$$

Why does this not work for an actual HMM?

Observation Probabilities

Observation Probs

Given $O = \{1, 3, 1, 2, 3, 1, \ldots\}$,
and the set of possible hidden states $\{H, C\}$,
find the T and B matrices.



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

Key: Expected number of times in state j

A Side Problem

You are the admissions coordinator for your department. You are trialing an office assistant robot that your colleague has requested you to test. You have to send offer letters to candidates who recently took the admission test. You ask the robot to do the task.

Unfortunately, there was an error in the visual perception so the robot randomly inserted the letters into envelopes (instead of the correct letter into the correct envelope) and posted them.

Out of the N accepted candidates, how many got their offer letter?

Letter Envelope Matching Problem

$$I_i = \begin{cases} 1 & \text{if the } i\text{th letter is in the correct envelope} \\ 0 & \text{otherwise} \end{cases}$$

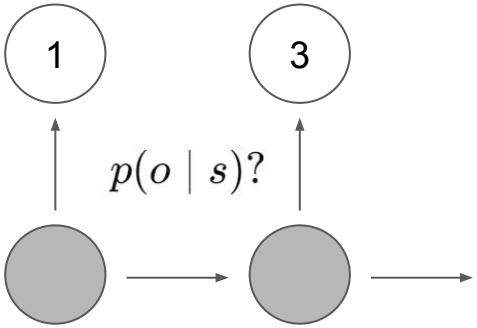
$$S = \sum_{i=1}^{n} I_i$$

Num correct placements

$$\mathbb{E}[S] = \mathbb{E}\left[\sum_{i=1}^{n} I_i\right] := \sum_{i=1}^{n} \mathbb{E}[I_i] \quad = \sum_{i=1}^{n} \frac{1}{n} = 1$$

Expected number of correct letters sent

Learning the Observation Prob.



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

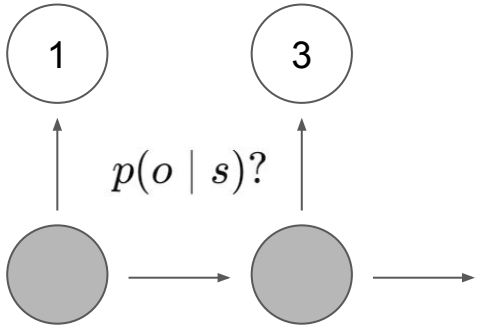
$$I_t = \begin{cases} 1 & \text{if } s_t = j \\ 0 & \text{otherwise} \end{cases}$$

$$S = \sum_{i=1}^{T} I_t$$

Number of times in state j over the sequence of T steps

$$\mathbb{E}[S] = \mathbb{E}\left[\sum_{i=1}^{T} I_t\right] = \sum_{i=1}^{T} \mathbb{E}\left[I_t\right] = \sum_{i=1}^{T} p(s_t = j)$$

Learning the Observation Prob.



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

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

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

How do we find gamma?

Finding Gamma

$$p(o_k \mid s_j) = \frac{\sum_{t=1}^{T} \gamma_t(j) 1(o_t = o_k)}{\sum_{t=1}^{T} \gamma_t(j)}$$

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

$$= \frac{p(s_t = j, o_{1:T})}{p(o_{1:T})}$$

$$p(o_{1:T}, s_t = j)$$

Numerator

Look familiar?

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

Backward Probability

$$p(o_{1:T}, s_t = j)$$

$$p(o_{1:T}, s_t = j) = p(o_{1:t}, s_t = j, o_{t+1:T})$$

$$\beta_t(i) = p(o_{t+1}, o_{t+2}, \ldots, o_T \mid s_t = i)$$

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

Why is this true?

Finding Backward Probability

$$\beta_t(i) = p(o_{t+1}, o_{t+2}, \ldots, o_T \mid s_t = i)$$

1. **Initialization:**

$$\beta_T(i) = 1, \quad 1 \leq i \leq N$$

2. **Recursion**

$$\beta_t(i) = \sum_{j=1}^{N} a_{ij}\, b_j(o_{t+1})\, \beta_{t+1}(j), \quad 1 \leq i \leq N, 1 \leq t < T$$

3. **Termination:**

$$P(O|\lambda) = \sum_{j=1}^{N} \pi_j\, b_j(o_1)\, \beta_1(j)$$

Source: SLP, Dan Jurafsky

Back to the Big Picture

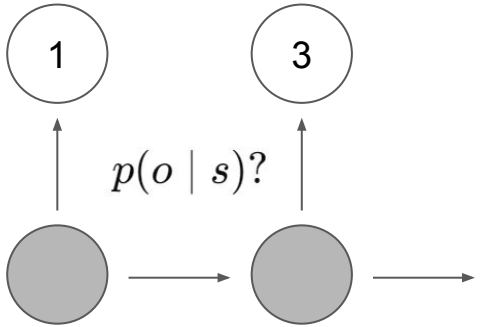
$$p(o_{1:T}, s_t = j) = \alpha_t(j) \cdot \beta_t(j)$$

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

$$= \frac{p(s_t = j, o_{1:T})}{p(o_{1:T})}$$

How do we find this?

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



Consolidation

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

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

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

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

Expected state occupancy count

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

Does this require knowing transition and observation?

Can't we just use alpha?

Consolidation

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

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

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

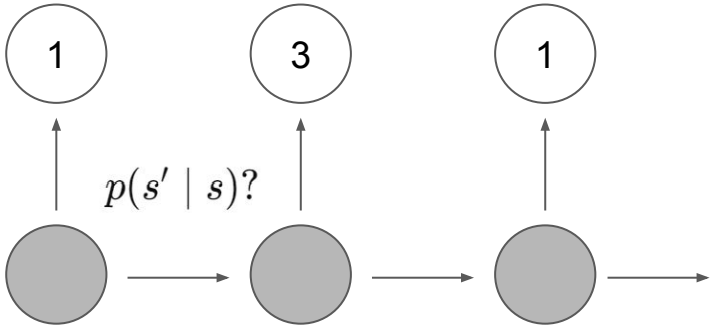
$$\gamma_t(j) = \frac{\alpha_t(j).\beta_t(j)}{\sum_{i=1}^{N} \alpha_T(i)}$$

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

Transition Probabilities

Transition Probability

Given $O = \{1, 3, 1, 2, 3, 1, \ldots\}$,
and the set of possible hidden states $\{H, C\}$,
find the T and B matrices.



$$p(s_{t+1} = j \mid s_t = i) = \frac{\text{expected number of transitions from state } i \text{ to state } j}{\text{expected number of transitions from state } i}$$

Learning the Transition Prob.

$$\frac{\text{expected number of transitions from state } i \text{ to state } j}{\text{expected number of transitions from state } i}$$

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

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

$$\text{almost-}\xi_t(i,j) = \alpha_t(i) \cdot a_{ij} \cdot b_j(o_{t+1}) \cdot \beta_{t+1}(j)$$

Why is this true?

$$\xi_t(i,j) = \frac{\text{almost-}\xi_t(i,j)}{p(o_{1:T})}$$

Learning the Transition Prob.

$$\frac{\text{expected number of transitions from state } i \text{ to state } j}{\text{expected number of transitions from state } i}$$

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

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

Why is this true?

Learning the Transition Prob.

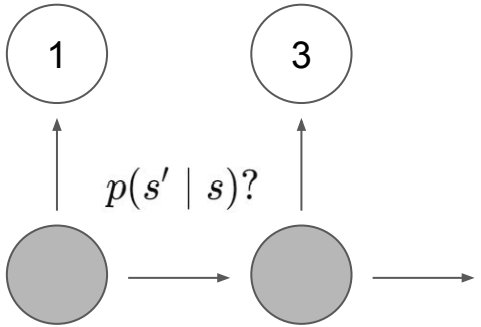
$$\frac{\text{expected number of transitions from state } i \text{ to state } j}{\text{expected number of transitions from state } i}$$

Overall flow?

$$\text{almost-}\xi_t(i,j) = \alpha_t(i) \cdot a_{ij} \cdot b_j(o_{t+1}) \cdot \beta_{t+1}(j)$$

$$\xi_t(i,j) = \frac{\text{almost-}\xi_t(i,j)}{p(o_{1:T})}$$

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



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

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

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

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

Expected state occupancy count

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

Expected state transition count

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

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

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

Why HMMs?

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

Modeling, inference, learning

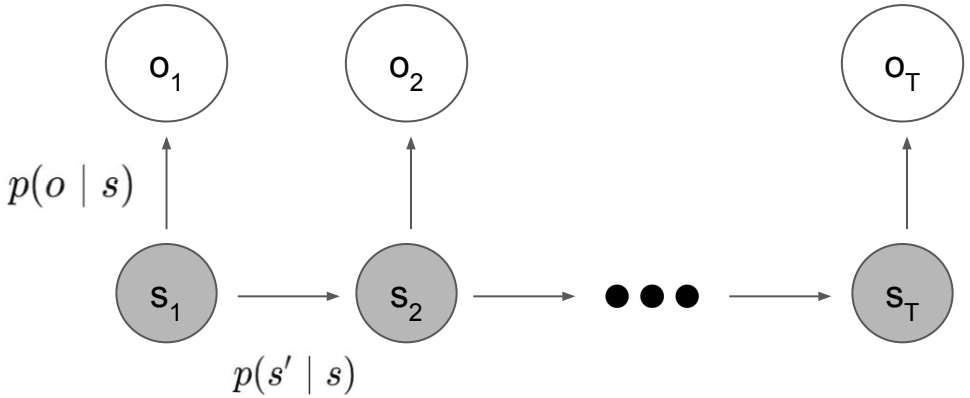
Viterbi

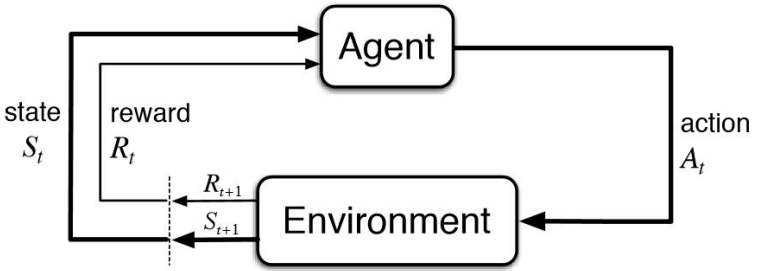
Source: Forbes

Source: USC

Markov Decision Processes

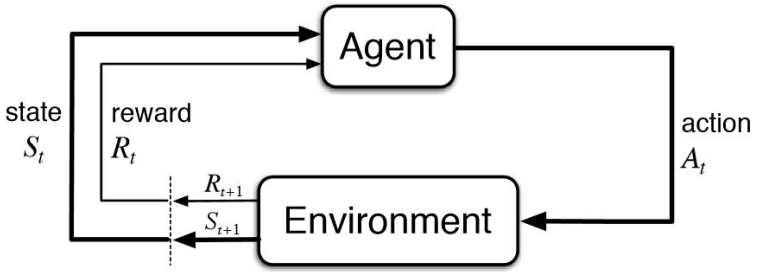
HMM: State Evolution





Source: Sutton & Barto

MDP Framework



Source: Sutton & Barto

- An abstraction for goal-directed behavior
- Whatever the details of sensors, memory and control
  - Any problem of learning goal-directed behavior can be reduced to three signals passing back and forth between an agent and its environment:
    - Represent choices made by the agent (the actions)
    - Represent basis on which choices are made (the states)
    - Define the agent's goal (the rewards)

About the Reward

- A way for you to specify what you want the agent to achieve…
    - NOT *how* you want it achieved

- The reward hypothesis

> That all of what we mean by goals and purposes can be well thought of as the maximization of the expected value of the cumulative sum of a received scalar signal (called reward).

From Reinforcement Learning: An Introduction, Sutton & Barto

MDP

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

$\mathcal{S}$: State Space

$\mathcal{A}$: Action Space

Difference from HMM?

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

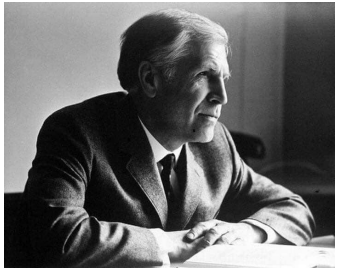
$R : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$: Reward Function

ρ : Initial State Distribution

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

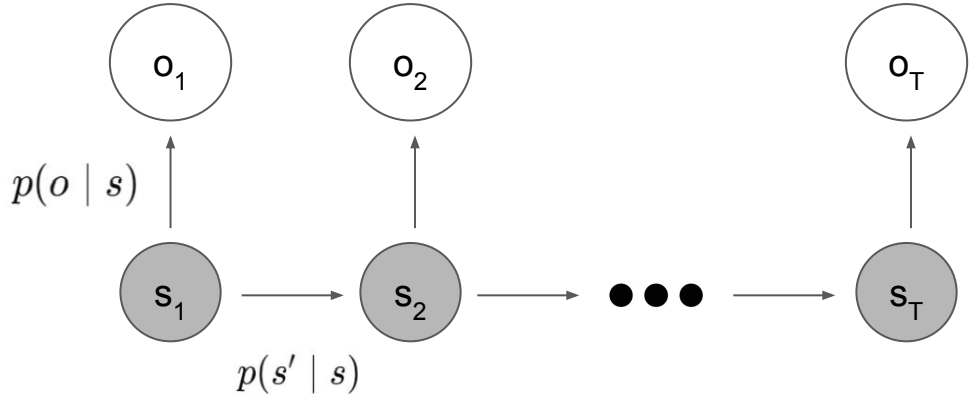
Richard Bellman, Source: Wikipedia

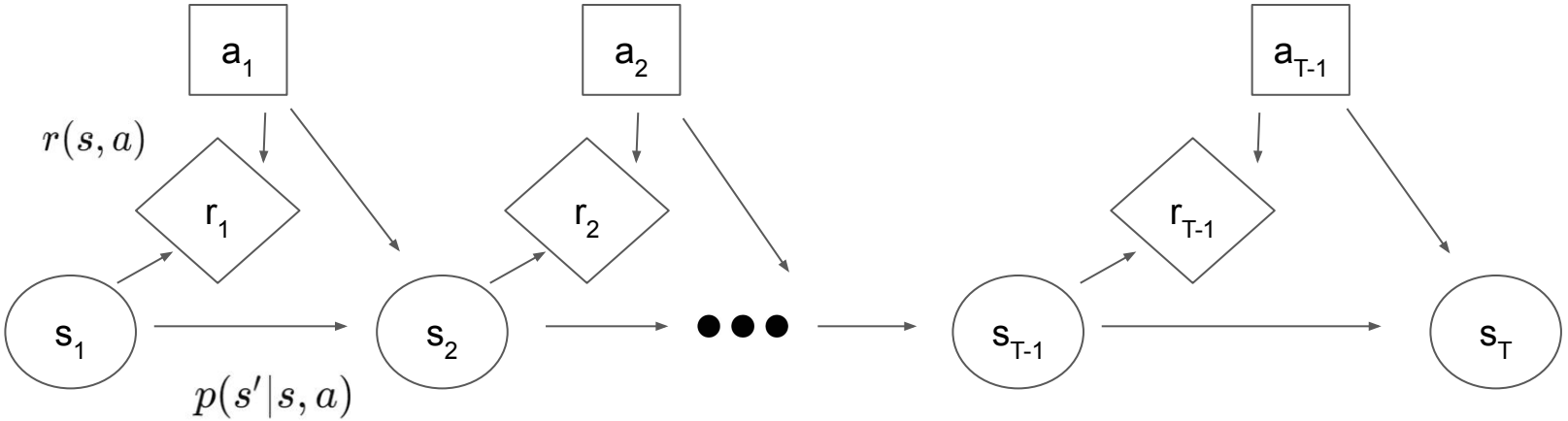
$x_t, u_t, c(x_t, u_t)$



Lev Pontryagin, Source: Wikipedia

MDP: State Evolution





Example Applications

- Cleaning Robot

Source: Youtube

Example Applications

- Cleaning Robot
- Walking Robot

Source: Youtube

Example Applications

- Cleaning Robot
- Walking Robot
- Pole balancing

Source: Youtube

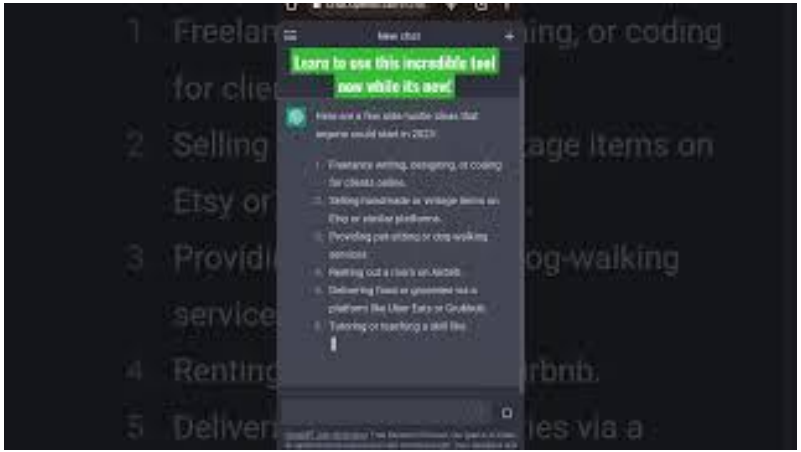
Example Applications

- Cleaning Robot
- Walking Robot
- Pole balancing
- Games: Tetris

Source: Youtube

Example Applications

- Cleaning Robot
- Walking Robot
- Pole balancing
- Games: Tetris
- Language: Dialog Systems



Source: Youtube

Example Applications

- Cleaning Robot
- Walking Robot
- Pole balancing
- Games: Tetris
- Language: Dialog Systems
- Computer Vision: Object Tracking

Source: Youtube

Example Applications

- Cleaning Robot
- Walking Robot
- Pole balancing
- Games: Tetris
- Language: Dialog Systems
- Computer Vision: Object Tracking
- Vision + Language: Image Captioning

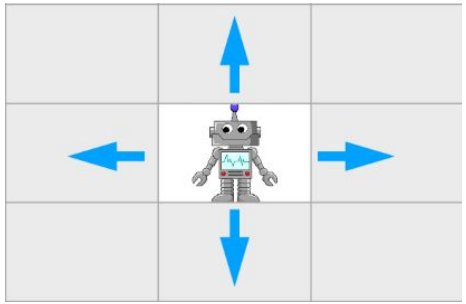
Source: Youtube

Example Applications

- Cleaning Robot
- Walking Robot
- Pole balancing
- Games: Tetris
- Language: Dialog Systems
- Computer Vision: Object Tracking
- Vision + Language: Image Captioning
- Server Management

Closer Look at Example MDPs

Grid World



- 10x10 grid
- Up, down, left, right
- 0.7 **correct** dir (as instructed), 0.1 rest
- Green cells are absorbing (end state)

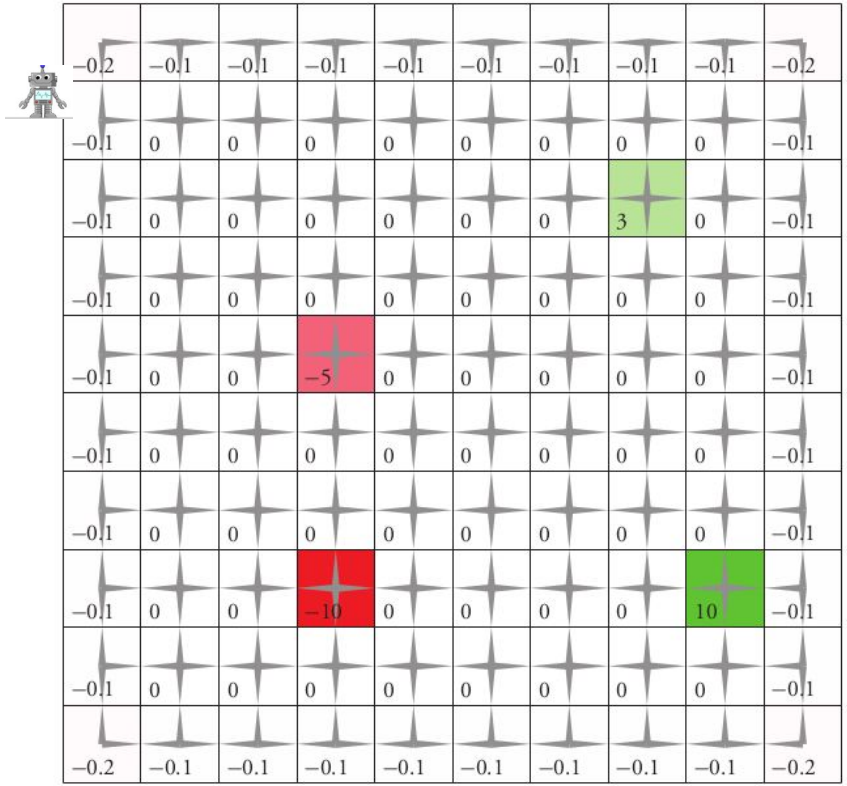
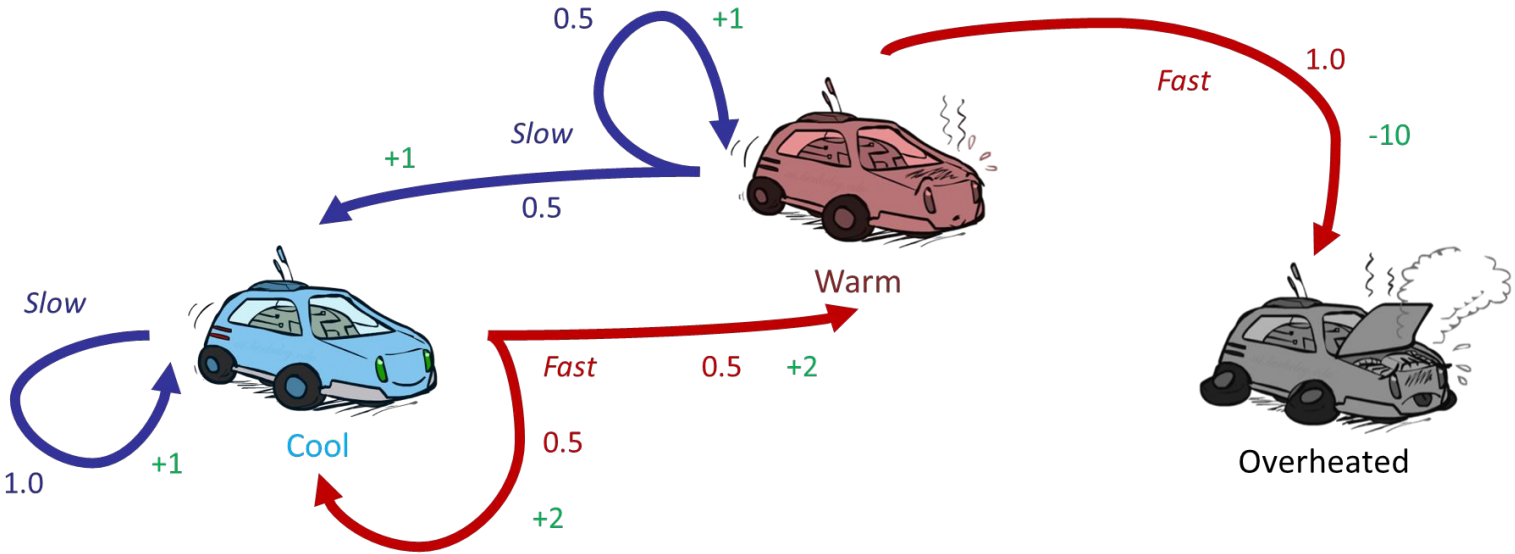


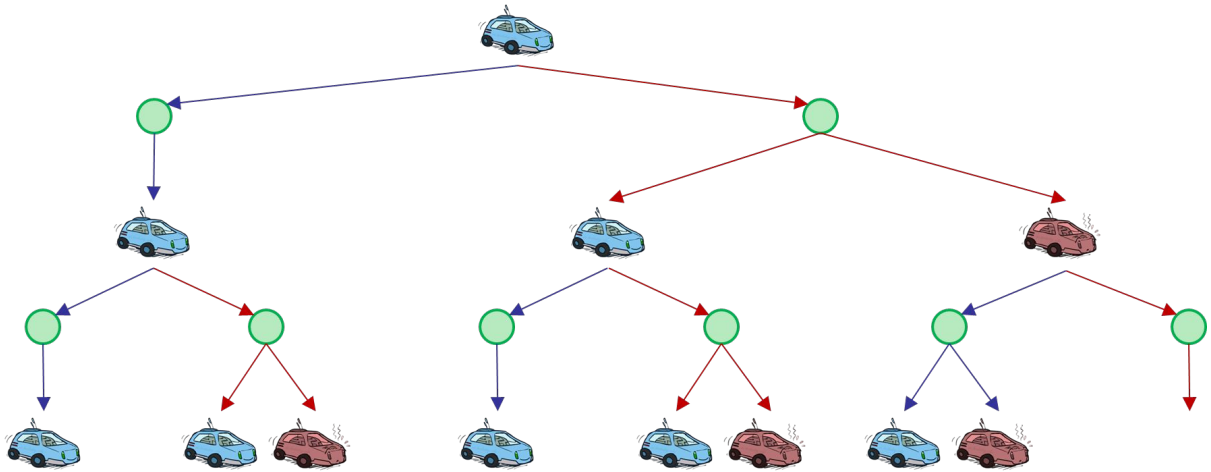
Figure from Algorithms for Decision Making, Mykel Kochenderfer

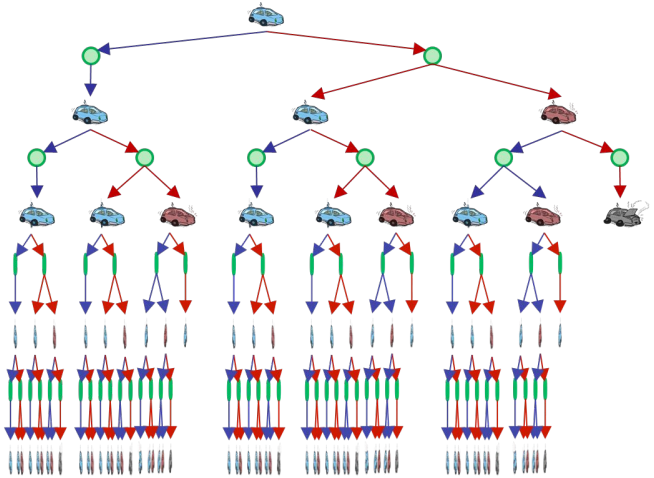
Racing Problem



From Xuesu Xiao

Racing Problem





Queuing Problem

Source: Dreamstime

- Customers line up in a queue. There is only one line. Line is empty initially
- We can serve one customer at a time. There are two modes of service: fast and slow
- Each timestep, a new customer arrives with probability p. The horizon length is T
- Waiting cost: gamma * queue length

Queuing Problem: Formulation

$\mathcal{S} = \{0, 1, 2, \ldots\}$: Length of the queue x_t $\quad x_0 = 0$

$\mathcal{U} = \{\text{Fast (F)}, \text{Slow (S)}\}$ $\quad$ Completion probs: $q(F) > q(S)$

$c(x_t, u_t) = \gamma x_t + d(u_t)$ $\quad$ Service costs: $d(F) > d(S)$

If $x = 0$:

$$p(x' = 1 \mid x = 0, u = F/S) = p$$

$$p(x' = 0 \mid x = 0, u = F/S) = 1 - p$$

If $x > 0$:

$$p(x' = x + 1 \mid x, u) = p \cdot (1 - q(u))$$

$$p(x' = x \mid x, u) = (1 - p) \cdot (1 - q(u)) + p \cdot q(u)$$

$$p(x' = x - 1 \mid x, u) = q(u) \cdot (1 - p)$$