Chapter 9: Partially Observable Markov Decision Problems

ROB311: Artificial Intelligence

Chandra Gummaluru

University of Toronto

Winter 2024

Reinforcement Learning (RL)

Recall that for an MDP with

Reinforcement Learning (RL)

Recall that for an MDP with

- transition probabilities, $p(\cdot|\cdot, \cdot)$

Reinforcement Learning (RL)

Recall that for an MDP with

- transition probabilities, $p(\cdot|\cdot,\cdot)$
- reward function, $r(\cdot,\cdot,\cdot)$

Reinforcement Learning (RL)

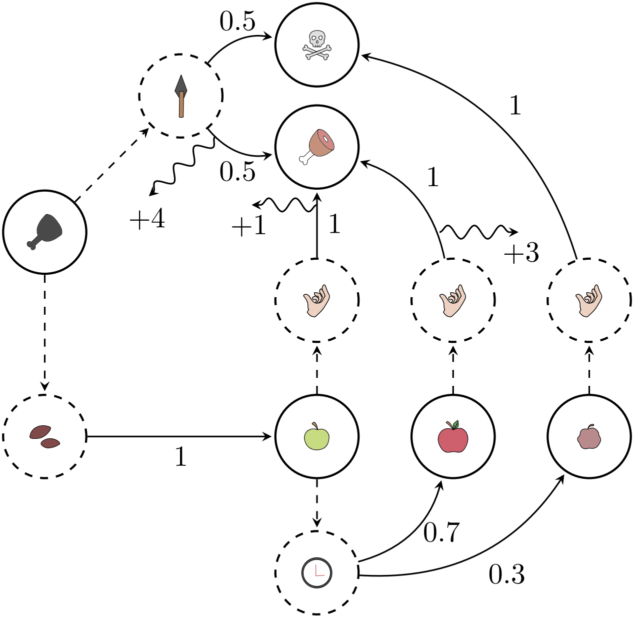
Recall that for an MDP with

- transition probabilities, $p(\cdot|\cdot,\cdot)$
- reward function, $r(\cdot,\cdot,\cdot)$

optimal quality function satisfied:

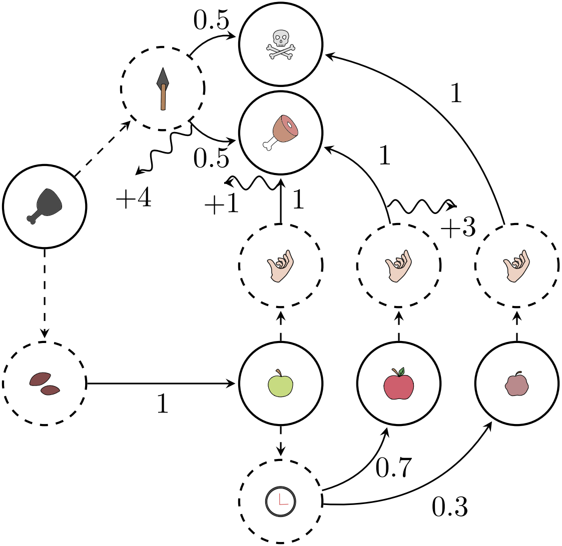
$$q^*(s,a) = \sum_{s'} p(s'|s,a)\left(r(s,a,s') + \gamma \max_{a'} q^*(s',a')\right).$$

Finite MDPs: Example



Finite MDPs: Example

Want to find optimal policy for  with $T = 3$ ($\gamma = 1$).



s	a	q^*	q^*
	$\varnothing$		
	$\varnothing$		

Reinforcement Learning (RL)

Recall that for an MDP with

- transition probabilities, $p(\cdot|\cdot,\cdot)$
- reward function, $r(\cdot,\cdot,\cdot)$

optimal quality function satisfied:

$$q^*(s,a) = \sum_{s'} p(s'|s,a)\left(r(s,a,s') + \gamma \max_{a'} q^*(s',a')\right).$$

Reinforcement Learning (RL)

Recall that for an MDP with

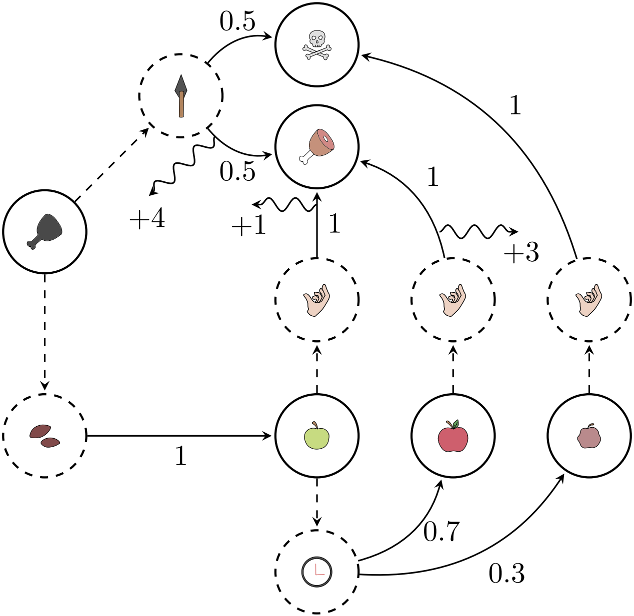
- transition probabilities, $p(\cdot|\cdot, \cdot)$
- reward function, $r(\cdot, \cdot, \cdot)$

optimal quality function satisfied:

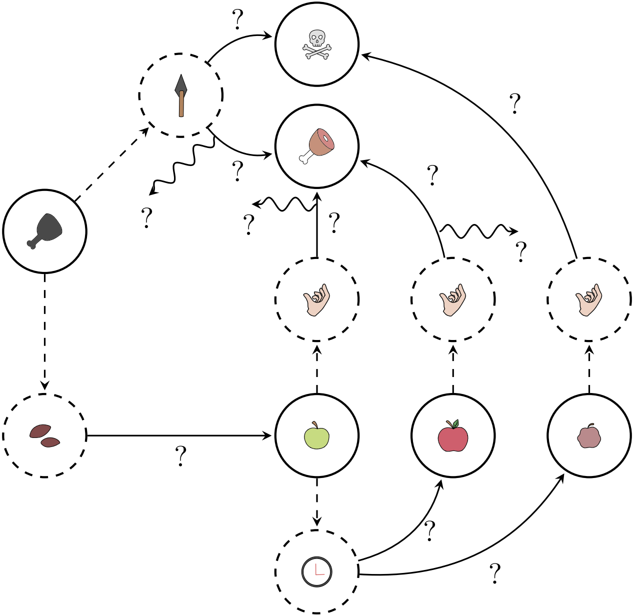
$$q^*(s, a) = \sum_{s'} p(s'|s, a) \left(r(s, a, s') + \gamma \max_{a'} q^*(s', a') \right).$$

Want to generalize when $p(\cdot|\cdot, \cdot)$ and/or $r(\cdot, \cdot, \cdot)$ unknown.

Finite MDPs: Example



Finite MDPs: Example



RL: Estimating q^* Empirically

If $\gamma = 0$, then:

RL: Estimating q^* Empirically

If $\gamma = 0$, then:

$$q^*(s,a) = \sum_{s'} p(s'|s,a) r(s,a,s'),$$

RL: Estimating q^* Empirically

If $\gamma = 0$, then:

$$q^*(s,a) = \sum_{s'} p(s'|s,a) r(s,a,s'),$$

i.e., $q^*(s,a)$ is expected value of immediate reward from s w/ a.

RL: Estimating q^* Empirically

If $\gamma = 0$, then:

$$q^*(s,a) = \sum_{s'} p(s'|s,a) r(s,a,s'),$$

i.e., $q^*(s,a)$ is expected value of immediate reward from s w/ a.

Thus, can estimate $q^*(s,a)$ as follows:

RL: Estimating q^* Empirically

If $\gamma = 0$, then:

$$q^*(s,a) = \sum_{s'} p(s'|s,a) r(s,a,s'),$$

i.e., $q^*(s,a)$ is expected value of immediate reward from s w/ a.

Thus, can estimate $q^*(s,a)$ as follows:

(1) simulate transition from s w/ a many (say K) times

RL: Estimating q^* Empirically

If $\gamma = 0$, then:

$$q^*(s,a) = \sum_{s'} p(s'|s,a) r(s,a,s'),$$

i.e., $q^*(s,a)$ is expected value of immediate reward from s w/ a.

Thus, can estimate $q^*(s,a)$ as follows:

(1) simulate transition from s w/ a many (say K) times

(2) compute empirical average reward

RL: Estimating q^* Empirically

If $\gamma = 0$, then:

$$q^*(s,a) = \sum_{s'} p(s'|s,a) r(s,a,s'),$$

i.e., $q^*(s,a)$ is expected value of immediate reward from s w/ a.

Thus, can estimate $q^*(s,a)$ as follows:

(1) simulate transition from s w/ a many (say K) times

(2) compute empirical average reward

$$\bar{R}_K = \frac{1}{K} \sum_{k=1}^{K} r_k,$$

where r_k is reward obtained in k^{th} simulation

RL: Estimating q^* Empirically

If $\gamma = 0$, then:

$$q^*(s,a) = \sum_{s'} p(s'|s,a) r(s,a,s'),$$

i.e., $q^*(s,a)$ is expected value of immediate reward from s w/ a.

Thus, can estimate $q^*(s,a)$ as follows:

(1) simulate transition from s w/ a many (say K) times
(2) compute empirical average reward

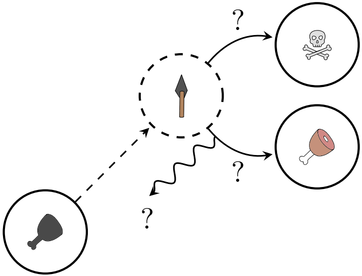
$$\bar{R}_K = \frac{1}{K} \sum_{k=1}^{K} r_k,$$

where r_k is reward obtained in k^{th} simulation

Can show that $\lim_{K \to \infty} \bar{R}_K = q^*(s,a)$.

RL: Estimating q^* Empirically

Consider simpler MDP:



RL: Estimating q^* Empirically

Simulate process from  many times:

RL: Estimating q^* Empirically

Simulate process from  many times:

$h_1 = \Big\langle \big(\varnothing, \varnothing,$  $, 0\big), \Big($  $,$  $,$  $, 4\Big)\Big\rangle$

RL: Estimating q^* Empirically

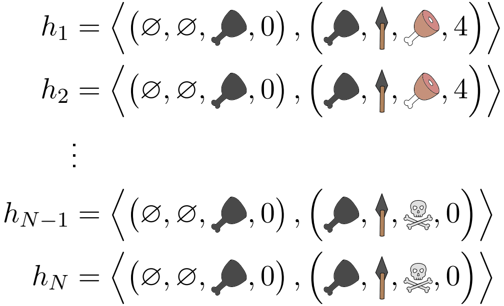
Simulate process from  many times:

$h_1 = \Big\langle \big(\varnothing, \varnothing,$  $, 0\big), \Big($  $,$  $,$  $, 4\Big)\Big\rangle$

$h_2 = \Big\langle \big(\varnothing, \varnothing,$  $, 0\big), \Big($  $,$  $,$  $, 4\Big)\Big\rangle$

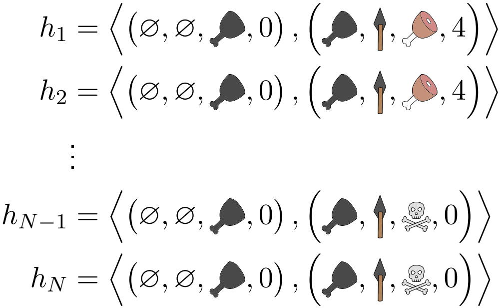
RL: Estimating q^* Empirically

Simulate process from  many times:



RL: Estimating q^* Empirically

Simulate process from  many times:

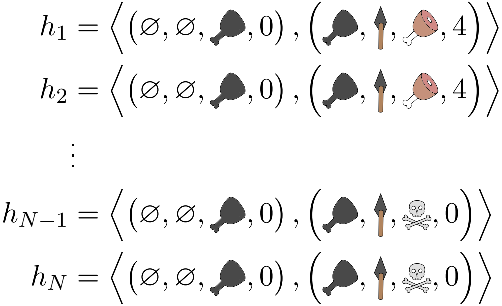


Compute empirical average:

$$\bar{R}_N = \frac{4 + 4 + \cdots + 2 + 2}{N}$$

RL: Estimating q^* Empirically

Simulate process from  many times:



Compute empirical average:

$$\bar{R}_N = \frac{4 + 4 + \cdots + 2 + 2}{N} \approx 2$$

Recursively Computing Averages

Average of $x_1, x_2, \ldots$:

Recursively Computing Averages

Average of $x_1, x_2, \ldots$:

$$\bar{x}_k = \frac{1}{k} \sum_{i=1}^{k} x_i$$

Recursively Computing Averages

Average of $x_1, x_2, \ldots$:

$$\begin{aligned}\bar{x}_k &= \frac{1}{k}\sum_{i=1}^{k} x_i \\ &= \frac{k-1}{k}\bar{x}_{k-1} + \frac{1}{k}x_k\end{aligned}$$

Recursively Computing Averages

Average of $x_1, x_2, \ldots$:

$$\begin{aligned}
\bar{x}_k &= \frac{1}{k}\sum_{i=1}^{k} x_i \\
&= \frac{k-1}{k}\bar{x}_{k-1} + \frac{1}{k}x_k \\
&= \bar{x}_{k-1} + \frac{1}{k}(x_k - \bar{x}_{k-1})
\end{aligned}$$

Recursively Computing Averages

Average of $x_1, x_2, \ldots$:

$$\begin{aligned}
\bar{x}_k &= \frac{1}{k}\sum_{i=1}^{k} x_i \\
&= \frac{k-1}{k}\bar{x}_{k-1} + \frac{1}{k}x_k \\
&= \bar{x}_{k-1} + \frac{1}{k}(x_k - \bar{x}_{k-1})
\end{aligned}$$

Update rule:

$$\bar{x} \leftarrow \bar{x} + \alpha\left(x_{\textsf{new}} - \bar{x}\right).$$

Recursively Computing Averages

Average of $x_1, x_2, \ldots$:

$$\begin{aligned}\bar{x}_k &= \frac{1}{k}\sum_{i=1}^{k} x_i \\ &= \frac{k-1}{k}\bar{x}_{k-1} + \frac{1}{k}x_k \\ &= \bar{x}_{k-1} + \frac{1}{k}(x_k - \bar{x}_{k-1})\end{aligned}$$

Update rule:

$$\bar{x} \leftarrow \bar{x} + \underset{\substack{\uparrow \\ \text{learning rate}}}{\alpha}\left(x_{\text{new}} - \bar{x}\right).$$

RL: Quality Function Update Rule

Apply to q function after each transition, (s, a, s'):

RL: Quality Function Update Rule

Apply to q function after each transition, (s, a, s'):

$$q^*(s,a) \leftarrow q^*(s,a) + \frac{1}{N(s,a)} \big(r(s,a,s') - q^*(s,a) \big)$$

RL: Quality Function Update Rule

Apply to q function after each transition, (s, a, s'):

$$q^*(s,a) \leftarrow q^*(s,a) + \frac{1}{N(s,a)} \big(r(s,a,s') - q^*(s,a) \big)$$

For $\gamma \neq 0$, use old q^* values:

RL: Quality Function Update Rule

Apply to q function after each transition, (s, a, s'):

$$q^*(s,a) \leftarrow q^*(s,a) + \frac{1}{N(s,a)} \big(r(s,a,s') - q^*(s,a)\big)$$

For $\gamma \neq 0$, use old q^* values:

$$q^*(s,a) \leftarrow q^*(s,a) + \frac{1}{N(s,a)} \left(\left[r(s,a,s') + \gamma \max_{a'} q^*(s',a') \right] - q^*(s,a) \right).$$

RL: Policy Extraction

To extract policy:

$$\pi(a|s) = \begin{cases} 1 & a = \arg\max_{a'} q^*(s, a) \\ 0 & \text{otherwise.} \end{cases}$$

RL: Estimating q^* Empirically

For original MDP, can simulate process from  w/ random actions:

RL: Estimating q^* Empirically

For original MDP, can simulate process from  w/ random actions:

$h = \langle (\varnothing, \varnothing,$  $, 0), ($  $,$  $,$  $, 0), ($  $,$  $,$  $, 0), ($  $,$  $,$  $, 3) \rangle$

RL: Q-Learning Algorithm

for each episode **do**
    set initial state $s \leftarrow s_0$
    while $s \notin \mathcal{T}$ **do**
        randomly choose an action in $\mathcal{A}(s)$
        get next state, s', and reward r
        update $N(s,a)$ and $q^*(s,a)$ as follows:

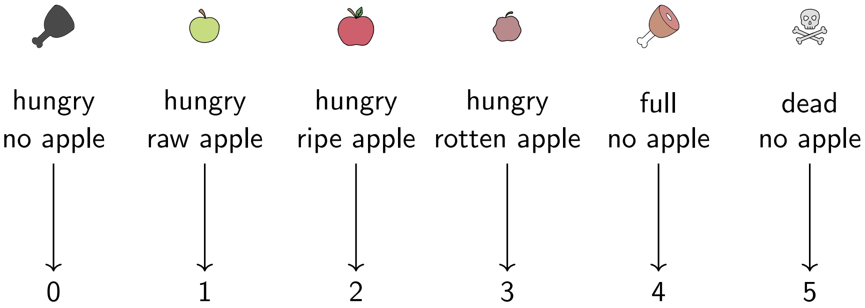
$$q^*(s,a) \leftarrow q^*(s,a) + \frac{1}{N(s,a)} \left(r(s,a,s') + \gamma \max_{a'} q^*(s',a') - q^*(s,a) \right)$$

$$N(s,a) \leftarrow N(s,a) + 1$$

        $s \leftarrow s'$
    end while
end for

RL: Q-Learning Code

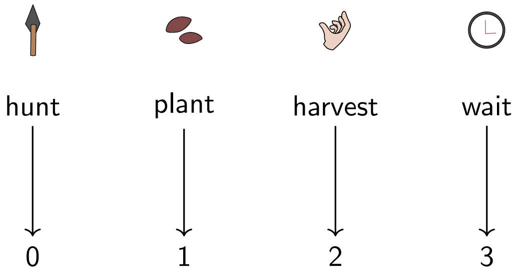
Form a bijection b/w state-space and subset of $\mathbb{N}$:




```
# define the state constants
HUNGRY, RAW, RIPE, ROTTEN, FULL, DEAD = 0, 1, 2, 3, 4, 5
states = [HUNGRY, RAW, RIPE, ROTTEN, FULL, DEAD]
```

RL: Q-Learning Code

Form a bijection b/w actions and subset of $\mathbb{N}$:




```python
# define the action constants
HUNT, PLANT, HARVEST, WAIT = 0, 1, 2, 3
actions = [HUNT, PLANT, HARVEST, WAIT]
```

RL: Q-Learning Code

Define the topology of the MDP:

- a map, `A`, so `A[s]` $\equiv A(s)$

```
# the set of legal actions from each state
A = {}
A[HUNGRY] = [HUNT, PLANT]
A[RAW]    = [HARVEST, WAIT]
A[RIPE]   = [HARVEST]
A[ROTTEN] = [HARVEST]
```

- a list, `tstates`, so `tstates` $\equiv \mathcal{T}$

```
tstates = [FULL, DEAD]
```

RL: Q-Learning Code

Define the properties of the MDP:

- a map, P, so P[s,a][s2] $\equiv p(s'|s,a)$

```
# the state transition distributions
P = {}
P[HUNGRY, HUNT]     = [0.0, 0.0, 0.0, 0.0, 0.5, 0.5]
P[HUNGRY, PLANT]    = [0.0, 1.0, 0.0, 0.0, 0.0, 0.0]
P[RAW, HARVEST]     = [0.0, 0.0, 0.0, 0.0, 1.0, 0.0]
P[RAW, WAIT]        = [0.0, 0.0, 0.7, 0.3, 0.0, 0.0]
P[RIPE, HARVEST]    = [0.0, 0.0, 0.0, 0.0, 1.0, 0.0]
P[ROTTEN, HARVEST] = [0.0, 0.0, 0.0, 0.0, 0.0, 1.0]
```

- a map, R, so R[s,a,s2] $\equiv r(s,a,s')$

```
# the reward function
R = {}
R[HUNGRY, HUNT, FULL]  = 4
R[RAW, HARVEST, FULL]  = 1
R[RIPE, HARVEST, FULL] = 3
```

RL: Q-Learning Code

Define the transition dynamics of the MDP:

```python
import numpy as np

def transition(s, a):
    # choose the next state based on the distribution
    snew = np.random.choice(states, p = P[s,a])

    # determine the reward
    if (s,a,snew) not in R.keys():
        rnew = 0
    else:
        rnew = R[(s,a,snew)]
    # return the next state and reward
    return snew, rnew
```

RL: Q-Learning Code

Define the Q-learning data:

- maps, `Q` and `N`, so `Q[s,a]` $\equiv q^*(s,a)$ and `N[s,a]` $\equiv N(s,a)$

```
Q, N = {}, {}
for s in states:
    for a in A[s]:
        Q[s,a] = 0
        N[s,a] = 0
```

- a scalar, `s0` $\equiv s_0$

```
s0 = HUNGRY
```

- a scalar, `sims`, denoting $\#$ of simulations

```
sims = 500
```

RL: Q-Learning Code

Define the Q-learning algorithm:

```
for t in range(1, sims):
    snew = s0

    while snew not in tstates:
      # randomly choose an action (and get next state).
      s = snew
      a = np.random.choice(A[s])
      N(s,a) += 1
      snew, rnew = transition(s, a)

      # find the maximum q-value from new state.
      qmax = 0
      if snew not in tstates:
        for a2 in A[snew]:
          qmax = max(qmax, Q[snew, a2])

      # apply q-learning update rule.
      Q[s,a] += 1/N[s,a] * (rnew + discount * qmax - Q[s,a])
```

RL: Training versus Testing

Episodes are classified as either

- training (sim): reward accumulated during episode does not count
- testing (test): reward accumulated during episode counts

Two common scenarios:

(1) K sims, 1 test

(2) K tests

RL: K-Sims, 1 Test

In the case of K sims, 1 test:

(1) select actions randomly during K sims

(2) extract optimal policy, π^*

(3) use π^* during test

RL: K tests

In the case of K tests:

- maximize average reward over K tests
- must balance between exploration and exploitation

Common ways to balance exploration and explotation:

(1) ε-greedy

(2) UCB

RL: K tests, ε-Greedy Algorithm

In episode k, choose optimal action w/ probability $\varepsilon(k)$, where:

RL: K tests, ε-Greedy Algorithm

In episode k, choose optimal action w/ probability $\varepsilon(k)$, where:

- $\varepsilon(0) \approx 0$

RL: K tests, ε-Greedy Algorithm

In episode k, choose optimal action w/ probability $\varepsilon(k)$, where:

- $\varepsilon(0) \approx 0$
- $\varepsilon(k)$ is increasing

RL: K tests, ε-Greedy Algorithm

In episode k, choose optimal action w/ probability $\varepsilon(k)$, where:

- $\varepsilon(0) \approx 0$
- $\varepsilon(k)$ is increasing
- $\varepsilon(k) \to 1$ as $k \to \infty$

RL: K tests, ε-Greedy Algorithm

In episode k, choose optimal action w/ probability $\varepsilon(k)$, where:

- $\varepsilon(0) \approx 0$
- $\varepsilon(k)$ is increasing
- $\varepsilon(k) \to 1$ as $k \to \infty$

Common choice for $\varepsilon(k)$ is $1 - 1/k$.

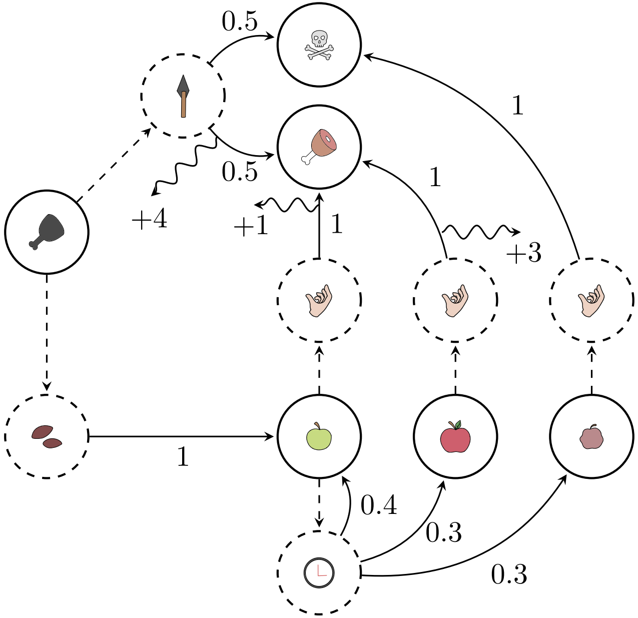
RL: K tests, UCB Algorithm

In episode k, choose action that maximizes $\text{UCB}(\cdot)$, where:

$$\text{UCB}(s,a) = \begin{cases} q^*(s,a) + C\sqrt{\frac{\log(k)}{N(s,a)}} & N(s,a) > 0 \\ \infty & \text{otherwise} \end{cases}$$

and $N(s,a) = \#$ of times a taken from s.

Infinite MDPs: Example



RL: Q-Learning Algorithm

for each episode **do**
  set initial state $s \leftarrow s_0$
  while $s \notin \mathcal{T}$ **do**
    randomly choose an action in $\mathcal{A}(s)$
    get next state, s', and reward r
    update $N(s,a)$ and $q^*(s,a)$ as follows:

$$q^*(s,a) \leftarrow q^*(s,a) + \frac{1}{N(s,a)}\left(r(s,a,s') + \gamma \max_{a'} q^*(s',a') - q^*(s,a)\right)$$

$$N(s,a) \leftarrow N(s,a) + 1$$

    $s \leftarrow s'$
  end while
end for

RL: Q-Learning Algorithm

for each episode **do**
  set initial state $s \leftarrow s_0$
  while $s \notin \mathcal{T}$ **do** $\longleftarrow$ possible infinite loop
    randomly choose an action in $\mathcal{A}(s)$
    get next state, s', and reward r
    update $N(s,a)$ and $q^*(s,a)$ as follows:

$$q^*(s,a) \leftarrow q^*(s,a) + \frac{1}{N(s,a)}\left(r(s,a,s') + \gamma \max_{a'} q^*(s',a') - q^*(s,a)\right)$$

$$N(s,a) \leftarrow N(s,a) + 1$$

    $s \leftarrow s'$
  end while
end for

RL: Q-Learning Algorithm

for each episode **do**
  $l \leftarrow 0$
  set initial state $s \leftarrow s_0$
  while $s \notin \mathcal{T}$ **and** $l < l_{\mathsf{max}}$ **do**
    randomly choose an action in $\mathcal{A}(s)$
    get next state, s', and reward r
    update $N(s,a)$ and $q^*(s,a)$ as follows:

$$q^*(s,a) \leftarrow q^*(s,a) + \frac{1}{N(s,a)}\left(r(s,a,s') + \gamma \max_{a'} q^*(s',a') - q^*(s,a)\right)$$

$$N(s,a) \leftarrow N(s,a) + 1$$

    $l \leftarrow l + 1$
    $s \leftarrow s'$
  end while
end for

Frame Title

Choice of γ and l_{max} are coupled:

- $\gamma \approx 1$ requires large l_{max}
- $\gamma \approx 0$ requires small l_{max}

Partially Observable MDPs (POMDPs)

Makes the following assumptions:

Partially Observable MDPs (POMDPs)

Makes the following assumptions:

- environment modelled using state space, $\mathcal{S}$

Partially Observable MDPs (POMDPs)

Makes the following assumptions:

- environment modelled using state space, $\mathcal{S}$
- single agent

Partially Observable MDPs (POMDPs)

Makes the following assumptions:

- environment modelled using state space, $\mathcal{S}$
- single agent
- $S_t =$ state after transition t

Partially Observable MDPs (POMDPs)

Makes the following assumptions:

- environment modelled using state space, $\mathcal{S}$
- single agent
- $S_t =$ state after transition t
- $A_t =$ action inducing transition t

Partially Observable MDPs (POMDPs)

Makes the following assumptions:

- environment modelled using state space, $\mathcal{S}$
- single agent
- $S_t =$ state after transition t
- $A_t =$ action inducing transition t
- stochastic state transitions w/ memoryless property:

$$S_T \perp S_0, A_1 \ldots, A_{T-1}, S_{T-2} | S_{T-1}, A_T$$

Partially Observable MDPs (POMDPs)

Makes the following assumptions:

- environment modelled using state space, $\mathcal{S}$
- single agent
- $S_t =$ state after transition t
- $A_t =$ action inducing transition t
- stochastic state transitions w/ memoryless property:

$$S_T \perp S_0, A_1 \ldots, A_{T-1}, S_{T-2} | S_{T-1}, A_T$$

- $R_t =$ reward for transition t, i.e., (S_{T-1}, A_T, S_T)

Partially Observable MDPs (POMDPs)

Makes the following assumptions:

- environment modelled using state space, $\mathcal{S}$
- single agent
- $S_t =$ state after transition t
- $A_t =$ action inducing transition t
- stochastic state transitions w/ memoryless property:

$$S_T \perp S_0, A_1 \ldots, A_{T-1}, S_{T-2} | S_{T-1}, A_T$$

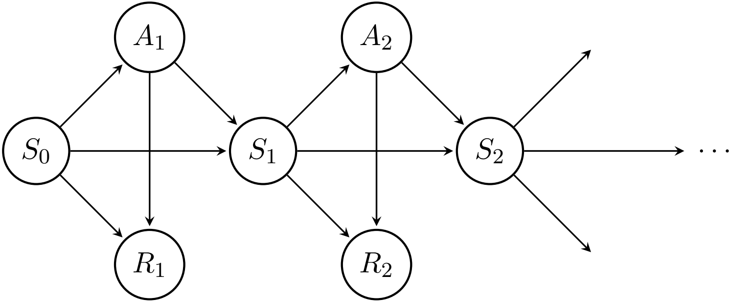
- $R_t =$ reward for transition t, i.e., (S_{T-1}, A_T, S_T)
- $O_t =$ observation of S_t

Markov Decision Processes (MDPs)

$S_0, A_1, R_1, S_1, A_2, R_2, S_2, \ldots$ form a Bayesian network:

Markov Decision Processes (MDPs)

$S_0, A_1, R_1, S_1, A_2, R_2, S_2, \ldots$ form a Bayesian network:

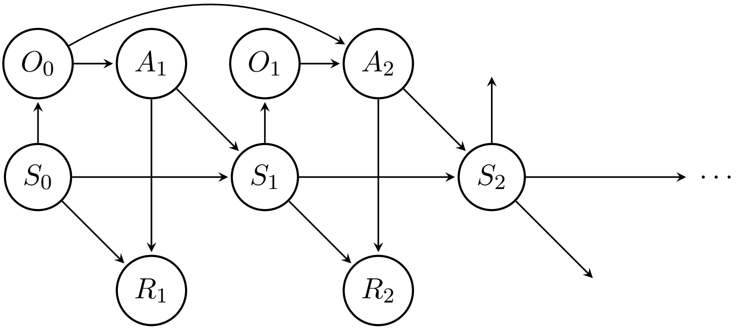


Partially Observable MDPs (POMDPs)

$S_0, O_0, A_1, R_1, S_1, O_1, A_2, R_2, S_2, O_2 \ldots$ form a Bayesian network:

Partially Observable MDPs (POMDPs)

$S_0, O_0, A_1, R_1, S_1, O_1, A_2, R_2, S_2, O_2 \ldots$ form a Bayesian network:



Markov Decision Processes (MDPs)

Need to specify…

Markov Decision Processes (MDPs)

Need to specify...

- initial state distribution:

$$p_0(s) := \mathbb{P}[S_0 = s]$$

Markov Decision Processes (MDPs)

Need to specify...

- initial state distribution:

$$p_0(s) := \mathbb{P}[S_0 = s]$$

- transition distribution:

$$p(s'|s,a) := \mathbb{P}[S_t = s'|A_t = a, S_{t-1} = s]$$

Markov Decision Processes (MDPs)

Need to specify...

- initial state distribution:

$$p_0(s) := \mathbb{P}[S_0 = s]$$

- transition distribution:

$$p(s'|s,a) := \mathbb{P}[S_t = s'|A_t = a, S_{t-1} = s]$$

- reward function:

$$r(s,a,s') := \text{reward for transition } (s,a,s')$$

Markov Decision Processes (MDPs)

Need to specify. . .

- initial state distribution:

$$p_0(s) := \mathbb{P}[S_0 = s]$$

- transition distribution:

$$p(s'|s, a) := \mathbb{P}[S_t = s'|A_t = a, S_{t-1} = s]$$

- reward function:

$$r(s, a, s') := \text{reward for transition } (s, a, s')$$

- policy for choosing actions:

$$\pi_t(a|o_0, \ldots, o_t) := \mathbb{P}[A_t = a|O_0 = o_0, \ldots, O_t = o_t]$$

Markov Decision Processes (MDPs)

Need to specify...

- initial state distribution:

$$p_0(s) := \mathbb{P}[S_0 = s]$$

- transition distribution:

$$p(s'|s,a) := \mathbb{P}[S_t = s'|A_t = a, S_{t-1} = s]$$

- reward function:

$$r(s,a,s') := \text{reward for transition } (s,a,s')$$

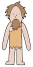
- policy for choosing actions:

$$\pi_t(a|o_0,\ldots,o_t) := \mathbb{P}[A_t = a|O_0 = o_0,\ldots,O_t = o_t]$$

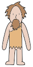
- measurement model:

$$m(o|s) := \mathbb{P}[O_t = o|S_t = s]$$

POMDPs: Measurement Models

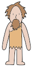
Now suppose  wants to feed child:

POMDPs: Measurement Models

Now suppose  wants to feed child:

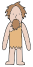
- cannot know satiety of child exactly

POMDPs: Measurement Models

Now suppose  wants to feed child:

- cannot know satiety of child exactly
- whether apple is edible or not must be inferred from senses

POMDPs: Measurement Models

Now suppose  wants to feed child:

- cannot know satiety of child exactly
- whether apple is edible or not must be inferred from senses

Possible observations for the apple:

Possible observations for the child's satiety:

:) :(:|

POMDPs: Measurement Models

Measurement distribution for child's satiety:

	:)	: (	: |
	0.0	0.8	0.2
	0.0	0.8	0.2
	0.0	0.8	0.2
	0.0	0.8	0.2
	0.8	0.2	0.0
	0.0	0.0	1.0

POMDPs: Measurement Models

Measurement distribution for the apple is:

					
	1.0	0.0	0.0	0.0	0.0
	0.2	0.6	0.2	0.0	0.0
	0.0	0.3	0.4	0.3	0.0
	0.0	0.0	0.0	0.2	0.8
	1.0	0.0	0.0	0.0	0.0
	1.0	0.0	0.0	0.0	0.0

Partially Observable MDPs (POMDPs)

Observe that policy is now time-dependent.

Partially Observable MDPs (POMDPs)

Observe that policy is now time-dependent.

If we assume agent cannot use past observations, i.e.,

$$A_t \perp O_0, \ldots, O_{t-1} | O_t,$$

Partially Observable MDPs (POMDPs)

Observe that policy is now time-dependent.

If we assume agent cannot use past observations, i.e.,

$$A_t \perp O_0, \ldots, O_{t-1} | O_t,$$

policy becomes time-independent,

$$\pi_t(a|o_0, \ldots, o_t) = \pi_0(a|o_t).$$

Partially Observable MDPs (POMDPs)

Observe that policy is now time-dependent.

If we assume agent cannot use past observations, i.e.,

$$A_t \perp O_0, \ldots, O_{t-1} | O_t,$$

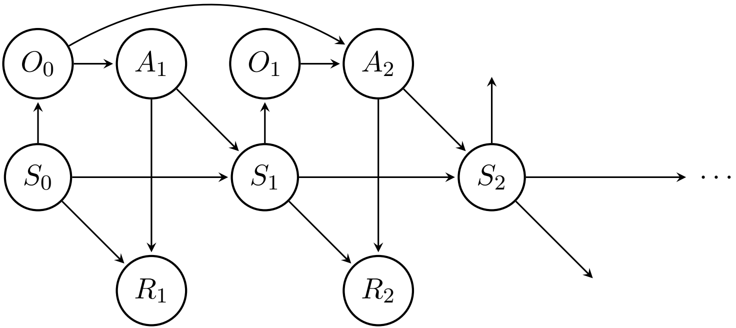
policy becomes time-independent,

$$\pi_t(a|o_0, \ldots, o_t) = \pi_0(a|o_t).$$

Only need to specify π_0.

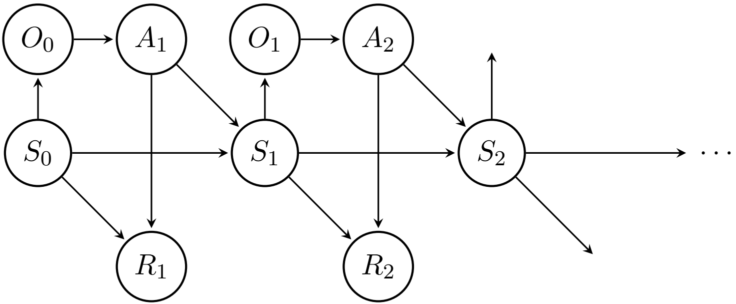
Partially Observable MDPs (POMDPs)

$S_0, O_0, A_1, R_1, S_1, O_1, A_2, R_2, S_2, O_2 \ldots$ form a Bayesian network:



Partially Observable MDPs (POMDPs)

$S_0, O_0, A_1, R_1, S_1, O_1, A_2, R_2, S_2, O_2 \ldots$ form a Bayesian network:



$\ldots$ assuming $A_t \perp O_0, \ldots, O_{t-1} | O_t$.

Partially Observable MDPs (POMDPs)

Observe that policy is now time-dependent.

If we assume agent cannot use past observations, i.e.,

$$A_t \perp O_0, \dots, O_{t-1} | O_t,$$

policy becomes time-independent,

$$\pi_t(a|o_0, \dots, o_t) = \pi_0(a|o_t).$$

Only need to specify π_0.

Partially Observable MDPs (POMDPs)

Observe that policy is now time-dependent.

If we assume agent cannot use past observations, i.e.,

$$A_t \perp O_0, \ldots, O_{t-1} | O_t,$$

policy becomes time-independent,

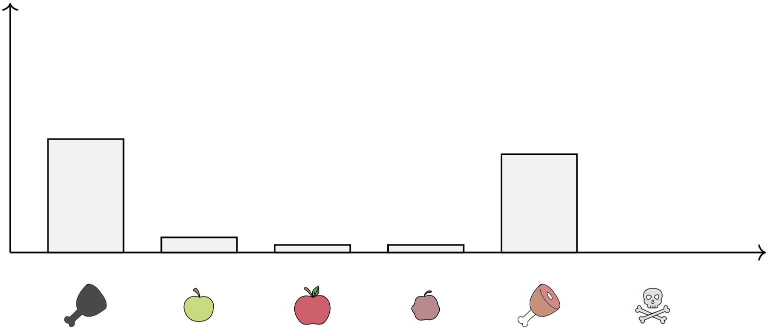
$$\pi_t(a|o_0, \ldots, o_t) = \pi_0(a|o_t).$$

Only need to specify π_0.

But assumption is not true in general.

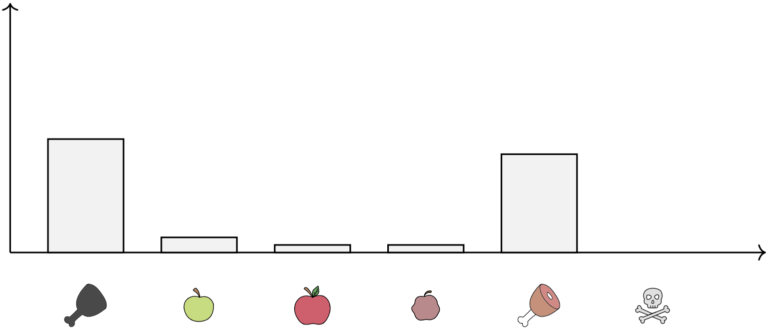
Partially Observable MDPs (POMDPs)

Maintain a belief (probability distribution) over the states:



Partially Observable MDPs (POMDPs)

Maintain a belief (probability distribution) over the states:



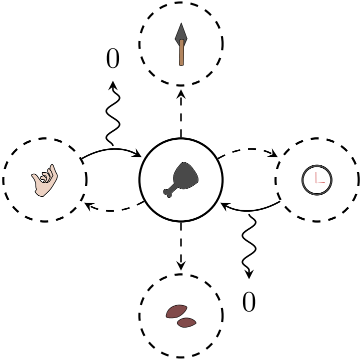
Assume actual state is the most likely state.

Partially Observable MDPs (POMDPs)

Since actual state is unknown, so are legal actions.

Can fix by assuming $\mathcal{A}(s) = \mathcal{A}(s') := \mathcal{A}$ for all s, s':

- if $a \notin \mathcal{A}(s)$, then $p(s'|s, a) = 0$ for all $s' \neq s$
- if $a \notin \mathcal{A}(s)$, then $r(s, a, s') = 0$ for all s'



Partially Observable MDPs (POMDPs)

Let b_t denote the belief after t observations:

Partially Observable MDPs (POMDPs)

Let b_t denote the belief after t observations:

- b_t is a probability distribution

Partially Observable MDPs (POMDPs)

Let b_t denote the belief after t observations:

- b_t is a probability distribution
- $b_t(s_t|a_{1:t}, o_{0:t}) = \mathbb{P}[S_t = s_t|A_t = a_t, O_{0:t} = o_{0:t}]$

Partially Observable MDPs (POMDPs)

Let b_t denote the belief after t observations:

- b_t is a probability distribution
- $b_t(s_t|a_{1:t}, o_{0:t}) = \mathbb{P}[S_t = s_t|A_t = a_t, O_{0:t} = o_{0:t}]$

Initial belief distribution is:

$$b_0(s_0) = \mathbb{P}[S_0 = s_0].$$

Partially Observable MDPs (POMDPs)

Let b_t denote the belief after t observations:

- b_t is a probability distribution
- $b_t(s_t|a_{1:t}, o_{0:t}) = \mathbb{P}[S_t = s_t|A_t = a_t, O_{0:t} = o_{0:t}]$

Initial belief distribution is:

$$b_0(s_0) = \mathbb{P}[S_0 = s_0].$$

Need a way to find $b_t(\cdot)$ from b_{t-1}, $a_{1:t}$, and $o_{0:t}$.

Partially Observable MDPs (POMDPs)

$$b_t(s_t|a_{1:t}, o_{0:t}) := \mathbb{P}[s_t|o_{0:t}, a_{1:t}]$$

Partially Observable MDPs (POMDPs)

$$
\begin{aligned}
b_t(s_t|a_{1:t}, o_{0:t}) &:= \mathbb{P}[s_t|o_{0:t}, a_{1:t}] \\
&= \eta \mathbb{P}[s_t, o_{0:t}, a_{1:t}]
\end{aligned}
$$

Partially Observable MDPs (POMDPs)

$$
\begin{aligned}
b_t(s_t|a_{1:t}, o_{0:t}) &:= \mathbb{P}[s_t|o_{0:t}, a_{1:t}] \\
&= \eta \mathbb{P}[s_t, o_{0:t}, a_{1:t}] \\
&= \eta \mathbb{P}[o_t|s_t] \mathbb{P}[s_t, o_{0:t-1}, a_{1:t}]
\end{aligned}
$$

Partially Observable MDPs (POMDPs)

$$
\begin{aligned}
b_t(s_t|a_{1:t}, o_{0:t}) &:= \mathbb{P}[s_t|o_{0:t}, a_{1:t}] \\
&= \eta \mathbb{P}[s_t, o_{0:t}, a_{1:t}] \\
&= \eta \mathbb{P}[o_t|s_t]\mathbb{P}[s_t, o_{0:t-1}, a_{1:t}] \\
&= \eta m(o_t|s_t) \sum_{s_{t-1}} \mathbb{P}[s_t, s_{t-1}, o_{0:t-1}, a_{1:t}]
\end{aligned}
$$

Partially Observable MDPs (POMDPs)

$$\begin{aligned}
b_t(s_t|a_{1:t}, o_{0:t}) &:= \mathbb{P}[s_t|o_{0:t}, a_{1:t}] \\
&= \eta \mathbb{P}[s_t, o_{0:t}, a_{1:t}] \\
&= \eta \mathbb{P}[o_t|s_t] \mathbb{P}[s_t, o_{0:t-1}, a_{1:t}] \\
&= \eta m(o_t|s_t) \sum_{s_{t-1}} \mathbb{P}[s_t, s_{t-1}, o_{0:t-1}, a_{1:t}] \\
&= \eta m(o_t|s_t) \sum_{s_{t-1}} \mathbb{P}[s_t|s_{t-1}, a_t] \mathbb{P}[s_{t-1}, o_{0:t-1}, a_{1:t}]
\end{aligned}$$

Partially Observable MDPs (POMDPs)

$$
\begin{aligned}
b_t(s_t|a_{1:t}, o_{0:t}) &:= \mathbb{P}[s_t|o_{0:t}, a_{1:t}] \\
&= \eta \mathbb{P}[s_t, o_{0:t}, a_{1:t}] \\
&= \eta \mathbb{P}[o_t|s_t] \mathbb{P}[s_t, o_{0:t-1}, a_{1:t}] \\
&= \eta m(o_t|s_t) \sum_{s_{t-1}} \mathbb{P}[s_t, s_{t-1}, o_{0:t-1}, a_{1:t}] \\
&= \eta m(o_t|s_t) \sum_{s_{t-1}} \mathbb{P}[s_t|s_{t-1}, a_t] \mathbb{P}[s_{t-1}, o_{0:t-1}, a_{1:t}] \\
&= \eta m(o_t|s_t) \sum_{s_{t-1}} p(s_t|s_{t-1}, a_t) \mathbb{P}[s_{t-1}, o_{0:t-1}, a_{1:t}]
\end{aligned}
$$

Partially Observable MDPs (POMDPs)

$$
\begin{aligned}
b_t(s_t|a_{1:t}, o_{0:t}) &:= \mathbb{P}[s_t|o_{0:t}, a_{1:t}] \\
&= \eta\mathbb{P}[s_t, o_{0:t}, a_{1:t}] \\
&= \eta\mathbb{P}[o_t|s_t]\mathbb{P}[s_t, o_{0:t-1}, a_{1:t}] \\
&= \eta m(o_t|s_t) \sum_{s_{t-1}} \mathbb{P}[s_t, s_{t-1}, o_{0:t-1}, a_{1:t}] \\
&= \eta m(o_t|s_t) \sum_{s_{t-1}} \mathbb{P}[s_t|s_{t-1}, a_t]\mathbb{P}[s_{t-1}, o_{0:t-1}, a_{1:t}] \\
&= \eta m(o_t|s_t) \sum_{s_{t-1}} p(s_t|s_{t-1}, a_t)\mathbb{P}[s_{t-1}, o_{0:t-1}, a_{1:t}] \\
&= m(o_t|s_t) \sum_{s_{t-1}} p(s_t|s_{t-1}, a_t)\mathbb{P}[s_{t-1}|o_{0:t-1}, a_{1:t-1}]
\end{aligned}
$$

Partially Observable MDPs (POMDPs)

$$
\begin{aligned}
b_t(s_t|a_{1:t}, o_{0:t}) &:= \mathbb{P}[s_t|o_{0:t}, a_{1:t}] \\
&= \eta\mathbb{P}[s_t, o_{0:t}, a_{1:t}] \\
&= \eta\mathbb{P}[o_t|s_t]\mathbb{P}[s_t, o_{0:t-1}, a_{1:t}] \\
&= \eta m(o_t|s_t)\sum_{s_{t-1}} \mathbb{P}[s_t, s_{t-1}, o_{0:t-1}, a_{1:t}] \\
&= \eta m(o_t|s_t)\sum_{s_{t-1}} \mathbb{P}[s_t|s_{t-1}, a_t]\mathbb{P}[s_{t-1}, o_{0:t-1}, a_{1:t}] \\
&= \eta m(o_t|s_t)\sum_{s_{t-1}} p(s_t|s_{t-1}, a_t)\mathbb{P}[s_{t-1}, o_{0:t-1}, a_{1:t}] \\
&= m(o_t|s_t)\sum_{s_{t-1}} p(s_t|s_{t-1}, a_t)\mathbb{P}[s_{t-1}|o_{0:t-1}, a_{1:t-1}] \\
&= m(o_t|s_t)\sum_{s_{t-1}} p(s_t|s_{t-1}, a_t)b_{t-1}(s_{t-1}|a_{1:t-1}, o_{1:t-1})
\end{aligned}
$$

Partially Observable MDPs (POMDPs)

Putting it all together:

$$b_t(s_t|a_{1:t}, o_{0:t}) = m(o_t|s_t) \sum_{s_{t-1}} p(s_t|s_{t-1}, a_t) b_{t-1}(s_{t-1}|a_{1:t-1}, o_{0:t-1}).$$

Partially Observable MDPs (POMDPs)

Putting it all together:

$$b_t(s_t|a_{1:t}, o_{0:t}) = m(o_t|s_t) \sum_{s_{t-1}} p(s_t|s_{t-1}, a_t) b_{t-1}(s_{t-1}|a_{1:t-1}, o_{0:t-1}).$$

Only holds for $t \geq 1$.

Partially Observable MDPs (POMDPs)

Putting it all together:

$$b_t(s_t|a_{1:t}, o_{0:t}) = m(o_t|s_t) \sum_{s_{t-1}} p(s_t|s_{t-1}, a_t) b_{t-1}(s_{t-1}|a_{1:t-1}, o_{0:t-1}).$$

Only holds for $t \geq 1$.

For $t = 0$ (assuming uniform prior):

$$b_0(s_0|o_0) = \frac{\mathbb{P}[o_0|s_0]\mathbb{P}[s_0]}{\sum_s \mathbb{P}[o_0|s]\mathbb{P}[s]}$$

Partially Observable MDPs (POMDPs)

Putting it all together:

$$b_t(s_t|a_{1:t}, o_{0:t}) = m(o_t|s_t) \sum_{s_{t-1}} p(s_t|s_{t-1}, a_t) b_{t-1}(s_{t-1}|a_{1:t-1}, o_{0:t-1}).$$

Only holds for $t \geq 1$.

For $t = 0$ (assuming uniform prior):

$$b_0(s_0|o_0) = \frac{\mathbb{P}[o_0|s_0]\mathbb{P}[s_0]}{\sum_s \mathbb{P}[o_0|s]\mathbb{P}[s]} = \frac{m(o_0|s_0)b_0(s_0)}{\sum_s m(o_0|s)b_0(s)}$$

Partially Observable MDPs (POMDPs)

Putting it all together:

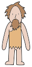
$$b_t(s_t|a_{1:t}, o_{0:t}) = m(o_t|s_t) \sum_{s_{t-1}} p(s_t|s_{t-1}, a_t) b_{t-1}(s_{t-1}|a_{1:t-1}, o_{0:t-1}).$$

Only holds for $t \geq 1$.

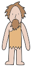
For $t = 0$ (assuming uniform prior):

$$b_0(s_0|o_0) = \frac{\mathbb{P}[o_0|s_0]\mathbb{P}[s_0]}{\sum_s \mathbb{P}[o_0|s]\mathbb{P}[s]} = \frac{m(o_0|s_0)b_0(s_0)}{\sum_s m(o_0|s)b_0(s)} = \frac{m(o_0|s_0)}{\sum_s m(o_0|s)}.$$

Partially Observable MDPs (POMDPs)

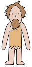
Assume the following for :

Partially Observable MDPs (POMDPs)

Assume the following for :

- initial distribution, $b_0(s_0)$ over states is uniform

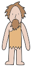
Partially Observable MDPs (POMDPs)

Assume the following for :

- initial distribution, $b_0(s_0)$ over states is uniform
- action sequence is

$\langle a_1, a_2, a_3 \rangle = \langle$ , ,  $\rangle$

Partially Observable MDPs (POMDPs)

Assume the following for :

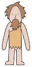
- initial distribution, $b_0(s_0)$ over states is uniform
- action sequence is

$\langle a_1, a_2, a_3 \rangle = \langle$ , ,  $\rangle$

- observation sequence is

$\langle o_0, o_1, o_2, o_3 \rangle = \Big\langle \Big(: (,$  $\Big), \Big(: (,$  $\Big), \Big(: (,$  $\Big), \Big(: |,$  $\Big) \Big\rangle$

Partially Observable MDPs (POMDPs)

Assume the following for :

- initial distribution, $b_0(s_0)$ over states is uniform
- action sequence is

$\langle a_1, a_2, a_3 \rangle = \langle$ , ,  $\rangle$

- observation sequence is

$\langle o_0, o_1, o_2, o_3 \rangle = \Big\langle \Big($:(,  $\Big), \Big($:(,  $\Big), \Big($:(,  $\Big), \Big($:|,  $\Big) \Big\rangle$

Want to find state distribution, $b_3(s_3 | a_{1:3}, o_{0:3})$.

Partially Observable MDPs (POMDPs)

Recall relevant equations:

$$b_0(s_0|o_0) = \frac{m(o_0|s_0)}{\sum_s m(o_0|s)}$$

and

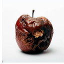
$$b_t(s_t|a_{1:t}, o_{0:t}) = m(o_t|s_t) \sum_{s_{t-1}} p(s_t|s_{t-1}, a_t) b_{t-1}(s_{t-1}|a_{1:t-1}, o_{0:t-1}).$$

Calculate the beliefs:

s	$b_0(s)$	$b_0(s\|o_0)$	$b_1(s\|o_{0:1}, a_1)$	$b_2(s\|o_{0:2}, a_{1:2})$	$b_3(s\|o_{0:3}, a_{1:3})$
					
					
					
					
					
					

POMDPs: Measurement Models

Measurement distributions are:

						:)	: (	: |
	1.0	0.0	0.0	0.0	0.0	0.0	0.8	0.2
	0.2	0.6	0.2	0.0	0.0	0.0	0.8	0.2
	0.0	0.3	0.4	0.3	0.0	0.0	0.8	0.2
	0.0	0.0	0.0	0.2	0.8	0.0	0.8	0.2
	1.0	0.0	0.0	0.0	0.0	0.0	0.8	0.2
	1.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0