CS221 Problem Workout

Week 4

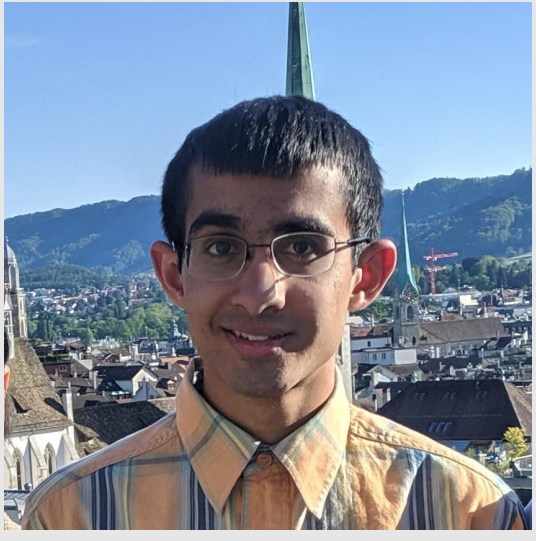
Stanford University

Introduction

Jenn Grannen

General OH: Mondays 9:30-11:00 Bytes + Zoom
HW OH: Fridays 9:30-11:00 Huang Basement

Chet Bhateja



HW OH: Mondays 6:30-7:30 PM Online
HW OH: Fridays 3:00-5:00 PM Huang Basement

Outline

- **MDPs as a model**
- What can we do with MDPs?
- Algorithms to solve problems related to MDPs
- Problem discussion

Search vs MDPs

- States
  - Start State
- Actions
- Goals
- Costs
- Successors

- States (S)
  - Start state
- Actions (A)
- ~~Goals~~
- Rewards (R)
- Transitions (T)
- *Discount factor (γ)*

MDPs (Markov Decision Processes) Modeling

- In **Markov decision processes**, agents take actions, move from state to state according to a transition function, and receive rewards along the way

Set of states S and set of actions A

- S may include initial and/or terminal states

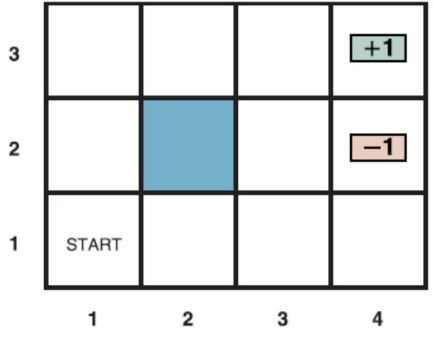
Transition function T: $S \times A \times S \rightarrow [0,1]$, where $T(s, a, s') = \Pr(s' | s, a)$

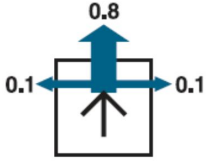
- Also called the model or dynamics of the problem

Reward function R: $S \times A \times S \rightarrow \mathbb{R}$, written as $R(s, a, s')$

Why is it called "Markov"? → Past states do not affect current decision making

MDPs Example





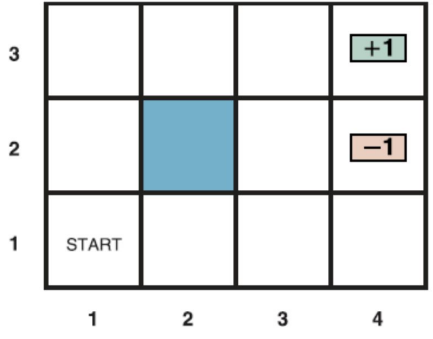
Example borrowed from Prof. Tony Dear, Columbia University Department of Computer Science.

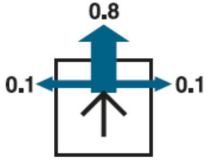
MDPs Example

States: Grid locations (11 in total)

- First two states in column 4 are terminal states

Actions: N, S, E, W (4 for each state)





Example borrowed from Prof. Tony Dear, Columbia University Department of Computer Science.

MDPs Example

States: Grid locations (11 in total)

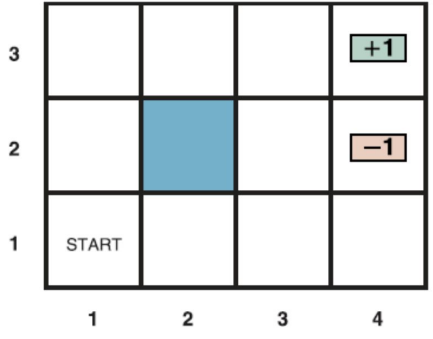
- First two states in column 4 are terminal states

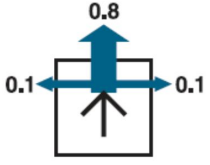
Actions: N, S, E, W (4 for each state)

Transition function: Intended direction with prob 0.8; otherwise, slip left or right with prob 0.1, respectively

- May stay in original state if moving into wall

Rewards: +1 or -1 for moving into terminal states; small negative reward otherwise (living reward)

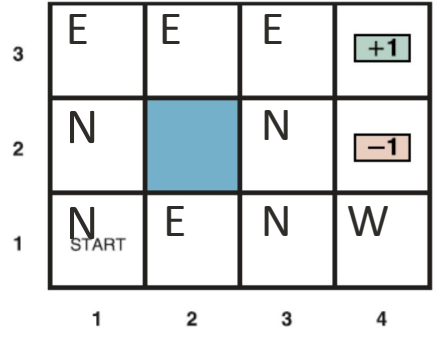


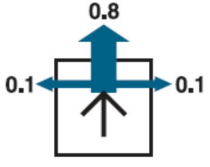


Example borrowed from Prof. Tony Dear, Columbia University Department of Computer Science.

What is a policy?

- Mapping from state to action, for example



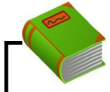


Outline

- MDPs as a model
- **What can we do with MDPs?**
  - **Evaluate a policy**
  - Find an optimal policy *while being able to exploit* - Life :)
- Algorithms to solve problems related to MDPs
- Problem discussion

1. Evaluate a given policy

- Given policy π, compute "how good" the policy is (aka value)
- The value of a policy at a state is the expected utility.



Definition: utility

Path: $s_0, a_1 r_1 s_1, a_2 r_2 s_2, \ldots$ (action, reward, new state).

The **utility** with discount γ is

$$u_1 = r_1 + \gamma r_2 + \gamma^2 r_3 + \gamma^3 r_4 + \cdots$$

$$V_\pi^{(t)}(s) \leftarrow \sum_{s'} T(s, \pi(s), s')[\text{Reward}(s, \pi(s), s') + \gamma V_\pi^{(t-1)}(s')]$$

- Related concept: Q value
  - $Q^\pi(s,a) = \Sigma\, T(s,\mathbf{a},s')\, [R(s,\mathbf{a},s') + \gamma\, V^\pi(s')]$

Outline

- MDPs as a model
- **What can we do with MDPs?**
  - Evaluate a policy
  - **Find an optimal policy** *while being able to exploit* - Life :)
- Algorithms to solve problems related to MDPs
- Problem discussion

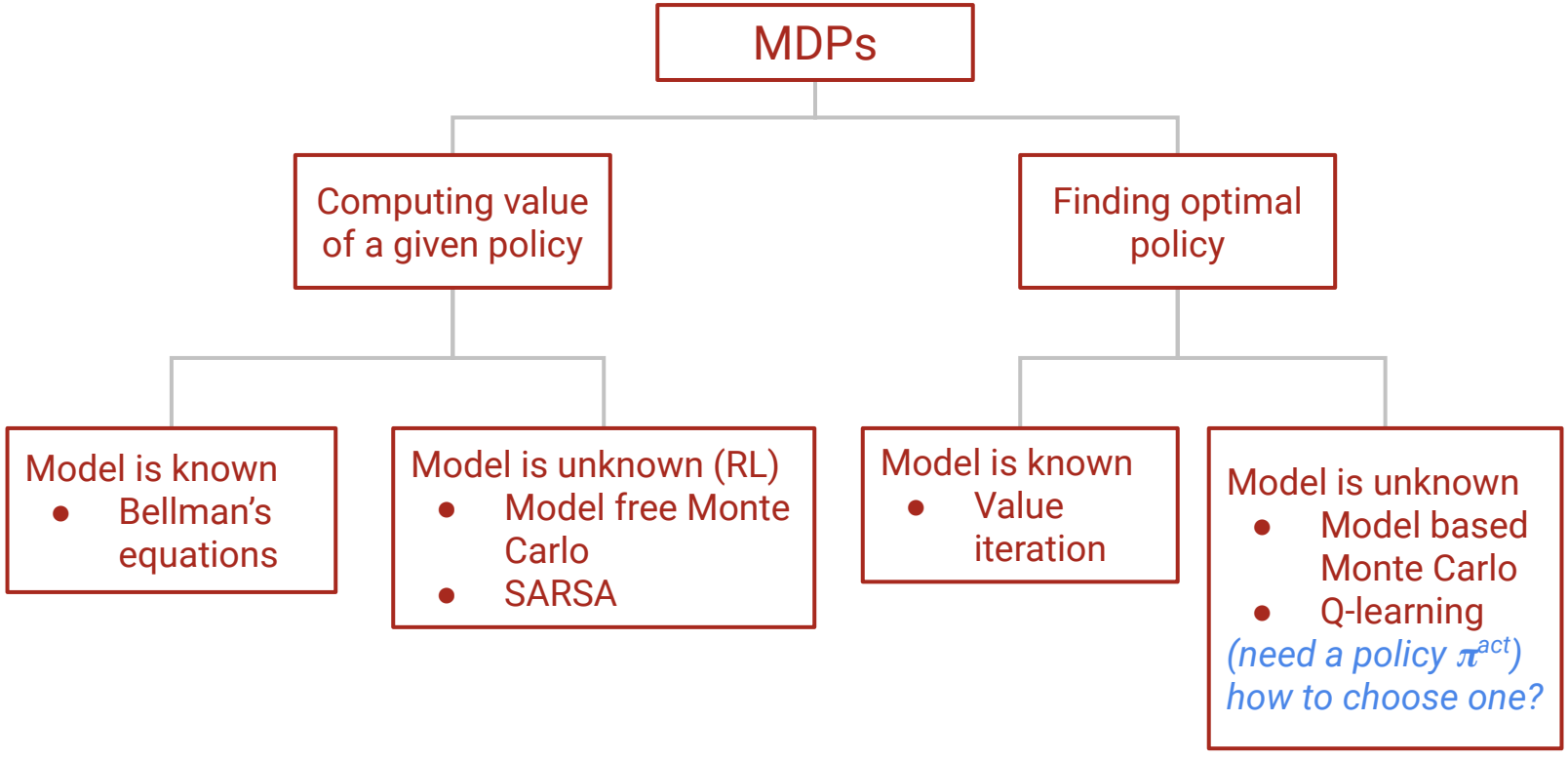
2. Find an **optimal** policy

- Given MDP M, find a policy π^* with highest value function V*
- i.e.
  - $V^*(s) >= V^{\pi}(s)$ … for all the states s

Outline

- MDPs as a model
- What can we do with MDPs?
  - Evaluate a policy
  - Find an optimal policy *while being able to exploit* - Life :)
- **Algorithms to solve problems related to MDPs**
- Problem discussion

Algorithms


MDPs
Computing value of a given policy
Finding optimal policy
Model is known
• Bellman's equations
Model is unknown (RL)
• Model free Monte Carlo
• SARSA
Model is known
• Value iteration
Model is unknown
• Model based Monte Carlo
• Q-learning
(need a policy π^{act})
how to choose one?

Algorithms

Evaluating a given policy π

Known model

Bellman's eqns

$$V_\pi^{(t)}(s) \leftarrow \underbrace{\sum_{s'} T(s,\pi(s),s')[\text{Reward}(s,\pi(s),s') + \gamma V_\pi^{(t-1)}(s')]}_{Q^{(t-1)}(s,\pi(s))}$$

Unknown model

Algorithm: SARSA

On each (s,a,r,s',a'):

$$\hat{Q}_\pi(s,a) \leftarrow (1-\eta)\hat{Q}_\pi(s,a) + \eta[\underbrace{r}_{\text{data}} + \gamma \underbrace{\hat{Q}_\pi(s',a')}_{\text{estimate}}]$$

Algorithm: model-free Monte Carlo

On each (s,a,u):

$$\hat{Q}_\pi(s,a) \leftarrow (1-\eta)\hat{Q}_\pi(s,a) + \eta \underbrace{u}_{\text{data}}$$

Computing optimal policy π^{opt}

Known model

Value iteration

$$V_{\text{opt}}^{(t)}(s) \leftarrow \underbrace{\max_{a \in \text{Actions}(s)} \sum_{s'} T(s,a,s')[\text{Reward}(s,a,s') + \gamma V_{\text{opt}}^{(t-1)}(s')]}_{Q_{\text{opt}}^{(t-1)}(s,a)}$$

Unknown model

Algorithm: Q-learning [Watkins/Dayan, 1992]

On each (s,a,r,s'):

$$\hat{Q}_{\text{opt}}(s,a) \leftarrow (1-\eta)\underbrace{\hat{Q}_{\text{opt}}(s,a)}_{\text{prediction}} + \eta\underbrace{(r + \gamma \hat{V}_{\text{opt}}(s'))}_{\text{target}}$$

Recall: $\hat{V}_{\text{opt}}(s') = \max_{a' \in \text{Actions}(s')} \hat{Q}_{\text{opt}}(s',a')$

Model based MC

$$\hat{T}(s,a,s') = \frac{\#\text{ times } (s,a,s') \text{ occurs}}{\#\text{ times } (s,a) \text{ occurs}}$$

$$\widehat{\text{Reward}}(s,a,s') = r \text{ in } (s,a,r,s')$$

How to choose policy π^{act}

- Greedy choice - choosing the optimal action (sub optimal result, as it does not explore enough)
- Random choice - estimates of Q values are good, but utility is bad
- Epsilon greedy - an interpolation between the two
  - Randomness allows for exploration
  - Exploitation leads to higher utility

Problems

Problem 1: MDP for Riding the Bus

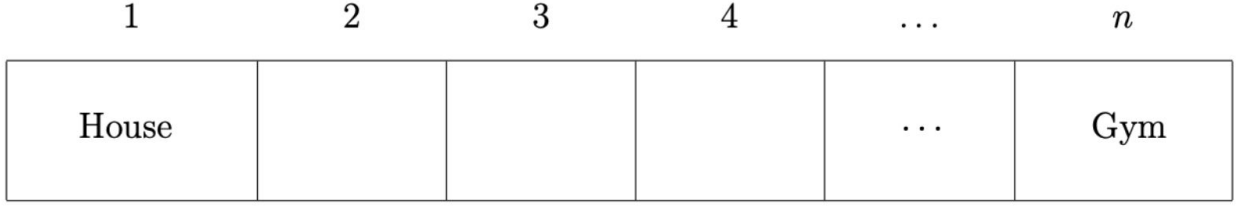
Problem 1: MDP for Riding the Bus

Identifying an MDP

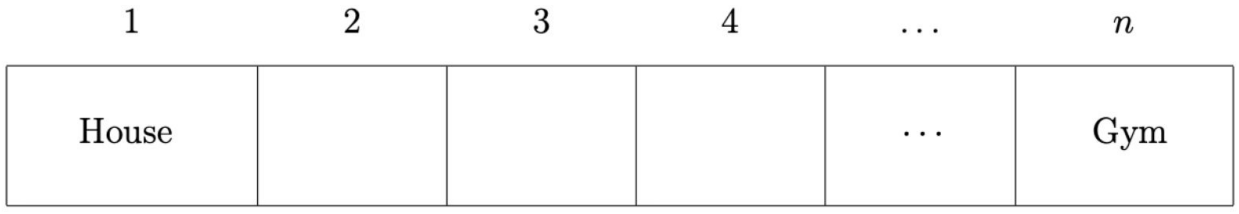
Identifying an MDP

Sabina wants to go from their house (located at location 1) to the gym (located at location n). At each location s, Sabina can either (i) deterministically walk forward to the next location $s+1$ (takes 1 unit of time) or (ii) wait for the bus.

The bus comes with probability ϵ, in which case, it will take Sabina to the gym in $1+\alpha(n-s)$ units of time, where α is some parameter. If the bus doesn't come, then Sabina stays put waiting for nothing, and that takes 1 unit of time.



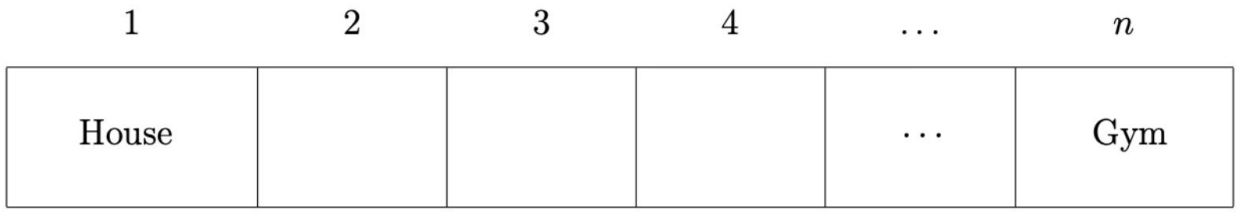
Identifying an MDP



How can we model this as an MDP?

- States
- Termination State
- Actions
- Rewards
- Transitions

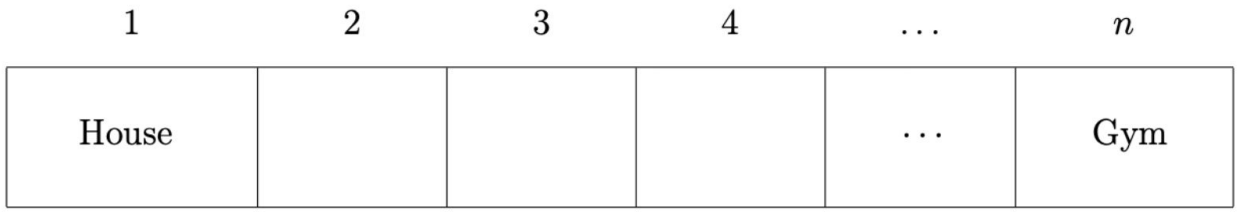
Identifying an MDP



How can we model this as an MDP?

- States:

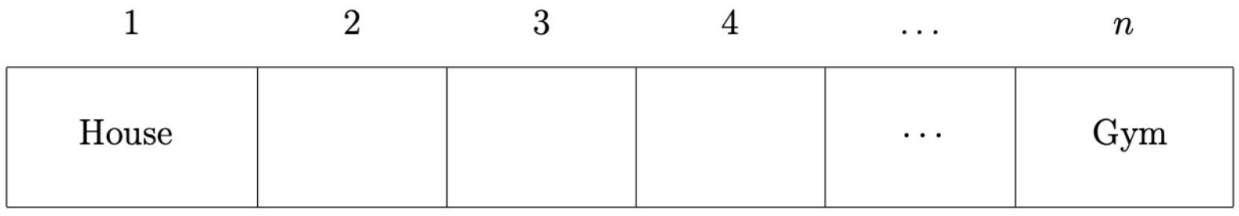
Identifying an MDP



How can we model this as an MDP?

- States: $s \in \{1, 2, \ldots, n\}$, Sabina's location.
- Termination State:

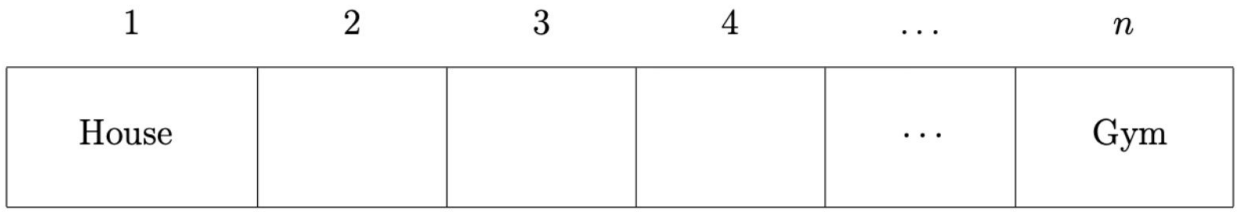
Identifying an MDP



How can we model this as an MDP?

- States: $s \in \{1, 2, \ldots, n\}$, Sabina's location.
- Termination State: $\mathbf{1}[s = n]$
- Actions:

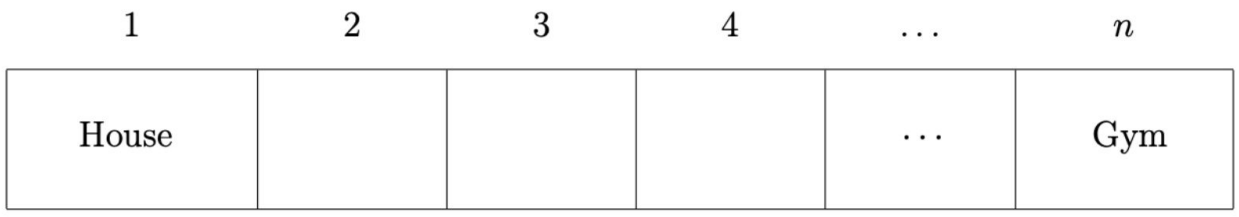
Identifying an MDP



How can we model this as an MDP?

- States: $s \in \{1, 2, \ldots, n\}$, Sabina's location.
- Termination State: $\mathbf{1}[s = n]$
- Actions: {Walk, Bus}
- Rewards:

Identifying an MDP



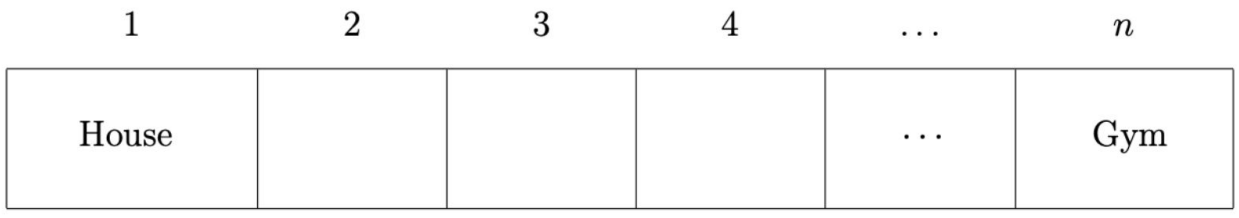
How can we model this as an MDP?

- Rewards:

$$\text{Reward}(s, \text{Walk}, s') = \begin{cases} -1 & \text{if } s' = s + 1 \\ -\infty & \text{otherwise} \end{cases}$$

$$\text{Reward}(s, \text{Bus}, s') = \begin{cases} -1 - \alpha(n - s) & \text{if } s' = n \\ -1 & \text{if } s' = s \\ -\infty & \text{otherwise} \end{cases}$$

Identifying an MDP



How can we model this as an MDP?

- Transitions:

$$T(s, \text{Walk}, s') = \begin{cases} 1 & \text{if } s' = s + 1 \\ 0 & \text{otherwise} \end{cases}$$

$$T(s, \text{Bus}, s') = \begin{cases} \epsilon & \text{if } s' = n \\ 1 - \epsilon & \text{if } s' = s \\ 0 & \text{otherwise} \end{cases}$$

Problem 1: MDP for Riding the Bus

Finding Policy Value

Finding Policy Value

Compute closed form expressions for (i) the value of a policy where Sabina always walks at every location and (ii) the value of a policy where Sabina always waits for the bus at every location. Assume a discount rate of $\gamma = 1$.

$$T(s, \text{Walk}, s') = \begin{cases} 1 & \text{if } s' = s + 1 \\ 0 & \text{otherwise} \end{cases} \qquad R(s, \text{Walk}, s') = \begin{cases} -1 & \text{if } s' = s + 1 \\ -\infty & \text{otherwise} \end{cases}$$

$$T(s, \text{Bus}, s') = \begin{cases} \epsilon & \text{if } s' = n \\ 1 - \epsilon & \text{if } s' = s \\ 0 & \text{otherwise} \end{cases} \qquad R(s, \text{Bus}, s') = \begin{cases} -1 - \alpha(n - s) & \text{if } s' = n \\ -1 & \text{if } s' = s \\ -\infty & \text{otherwise} \end{cases}$$

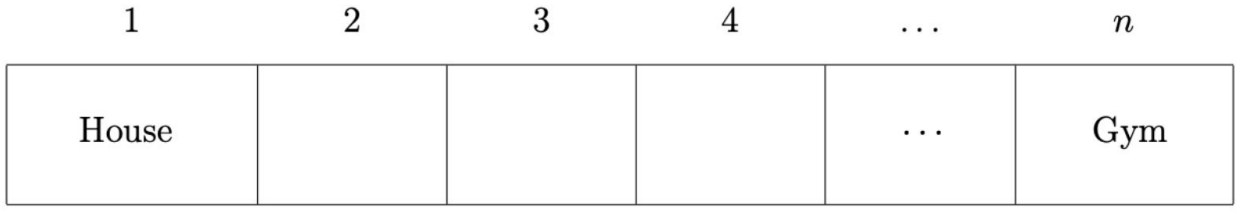
$$V_{\text{Walk}}(s) = \_\_\_\_\_$$

$$V_{\text{Bus}}(s) = \_\_\_\_\_$$

Finding Policy Value

Compute closed form expressions for the value of a policy where Sabina always walks at every location.

$$T(s, \text{Walk}, s') = \begin{cases} 1 & \text{if } s' = s + 1 \\ 0 & \text{otherwise} \end{cases} \qquad R(s, \text{Walk}, s') = \begin{cases} -1 & \text{if } s' = s + 1 \\ -\infty & \text{otherwise} \end{cases}$$

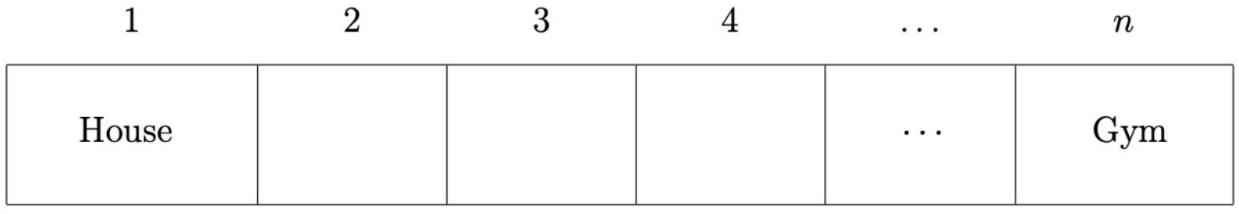


Note: The transition function for walking is deterministic!

Finding Policy Value

Compute closed form expressions for the value of a policy where Sabina always walks at every location.

$$T(s, \text{Walk}, s') = \begin{cases} 1 & \text{if } s' = s + 1 \\ 0 & \text{otherwise} \end{cases} \qquad R(s, \text{Walk}, s') = \begin{cases} -1 & \text{if } s' = s + 1 \\ -\infty & \text{otherwise} \end{cases}$$



Note: The transition function for walking is deterministic!

$$V_{\text{Walk}}(s) = -(n - s)$$

Finding Policy Value

Compute closed form expressions for the value of a policy where Sabina always waits for the bus at every location. Assume a discount rate of $\gamma = 1$.

$$T(s, \text{Bus}, s') = \begin{cases} \epsilon & \text{if } s' = n \\ 1 - \epsilon & \text{if } s' = s \\ 0 & \text{otherwise} \end{cases} \qquad R(s, \text{Bus}, s') = \begin{cases} -1 - \alpha(n - s) & \text{if } s' = n \\ -1 & \text{if } s' = s \\ -\infty & \text{otherwise} \end{cases}$$

Finding Policy Value

Compute closed form expressions for the value of a policy where Sabina always waits for the bus at every location. Assume a discount rate of $\gamma = 1$.

$$T(s, \text{Bus}, s') = \begin{cases} \epsilon & \text{if } s' = n \\ 1 - \epsilon & \text{if } s' = s \\ 0 & \text{otherwise} \end{cases} \qquad R(s, \text{Bus}, s') = \begin{cases} -1 - \alpha(n - s) & \text{if } s' = n \\ -1 & \text{if } s' = s \\ -\infty & \text{otherwise} \end{cases}$$

The transition function this time is NOT deterministic! Need:

$$V_\pi(s) = \sum_{s'} T(s, \pi(s), s') \left[\text{Reward}(s, \pi(s), s') + \gamma V_\pi(s')\right]$$

Finding Policy Value

Compute closed form expressions for the value of a policy where Sabina always waits for the bus at every location. Assume a discount rate of $\gamma = 1$.

$$T(s, \text{Bus}, s') = \begin{cases} \epsilon & \text{if } s' = n \\ 1 - \epsilon & \text{if } s' = s \\ 0 & \text{otherwise} \end{cases} \qquad R(s, \text{Bus}, s') = \begin{cases} -1 - \alpha(n - s) & \text{if } s' = n \\ -1 & \text{if } s' = s \\ -\infty & \text{otherwise} \end{cases}$$

The transition function this time is NOT deterministic! Need:

$$V_\pi(s) = \sum_{s'} T(s, \pi(s), s') \left[\text{Reward}(s, \pi(s), s') + \gamma V_\pi(s')\right]$$

$$V_{\text{Bus}}(s) = \mathbb{P}[\text{bus comes and we go to } n](R(s, \text{Bus}, n) + \gamma V_{\text{Bus}}(n))$$
$$+\mathbb{P}[\text{bus doesn't come and we stay at } s](R(s, \text{Bus}, s) + \gamma V_{\text{Bus}}(s))$$

Finding Policy Value

Compute closed form expressions for the value of a policy where Sabina always waits for the bus at every location. Assume a discount rate of $\gamma = 1$.

$$T(s, \text{Bus}, s') = \begin{cases} \epsilon & \text{if } s' = n \\ 1 - \epsilon & \text{if } s' = s \\ 0 & \text{otherwise} \end{cases} \qquad R(s, \text{Bus}, s') = \begin{cases} -1 - \alpha(n - s) & \text{if } s' = n \\ -1 & \text{if } s' = s \\ -\infty & \text{otherwise} \end{cases}$$

$$V_{\text{Bus}}(s) = \mathbb{P}[\text{bus comes and we go to } n](R(s, \text{Bus}, n) + \gamma V_{\text{Bus}}(n))$$
$$+\mathbb{P}[\text{bus doesn't come and we stay at } s](R(s, \text{Bus}, s) + \gamma V_{\text{Bus}}(s))$$

$$V_{\text{Bus}}(s) = \epsilon(-1 - \alpha(n - s)) + (1 - \epsilon)(-1 + V_{\text{Bus}}(s))$$
$$= -\alpha(n - s) - \frac{1}{\epsilon}$$

Finding Policy Value

For what values of ϵ (as a function of α and n) is it advantageous to walk rather than take the bus?

$$V_{\text{Walk}}(s) = -(n-s) \qquad V_{\text{Bus}}(s) = -\alpha(n-s) - \frac{1}{\epsilon}$$

For walking to be preferable we need:

Finding Policy Value

For what values of ϵ (as a function of α and n) is it advantageous to walk rather than take the bus?

$$V_{\text{Walk}}(s) = -(n-s) \qquad V_{\text{Bus}}(s) = -\alpha(n-s) - \frac{1}{\epsilon}$$

For walking to be preferable we need: $V_{\text{Walk}}(s) \geq V_{\text{Bus}}(s)$:

Finding Policy Value

For what values of ϵ (as a function of α and n) is it advantageous to walk rather than take the bus?

$$V_{\text{Walk}}(s) = -(n-s) \qquad V_{\text{Bus}}(s) = -\alpha(n-s) - \frac{1}{\epsilon}$$

For walking to be preferable we need: $V_{\text{Walk}}(s) \geq V_{\text{Bus}}(s)$:

$$n - s \leq \alpha(n-s) + \frac{1}{\epsilon}$$

$$(1-\alpha)(n-s) \leq \frac{1}{\epsilon}$$

Which leads to:

$$\begin{cases} \epsilon \leq \frac{1}{(1-\alpha)(n-s)} & \alpha < 1 \\ \epsilon > 0 & \alpha \geq 1 \end{cases}$$

Problem 1: MDP for Riding the Bus

Q-Learning

Q-Learning

Unfortunately, Sabina's town is unable to provide transition probabilities or a reward function (i.e. a bus schedule), making the above MDP possibly (and likely) inaccurate. To get around this, Sabina decides to use reinforcement learning, specifically Q-learning to determine the best policy. Sabina starts going around town both by bus and by walking, recording the following data:

s_0	a_1	r_1	s_1	a_2	r_2	s_2	a_3	r_3	s_3	a_4	r_4	s_4	a_5	r_5	s_5
1	Bus	-1	1	Bus	-1	1	Bus	3	3	Walk	1	4	Walk	1	5

Q-Learning

Run the Q-learning algorithm once over the given data to compute an estimate of the optimal Q-value $Q_{\text{opt}}(s, a)$. Process the episodes from left to right. Assume all Q-values are initialized to zero, and use a learning rate of $\eta = 0.5$ and a discount of $\gamma = 1$.

s_0	a_1	r_1	s_1	a_2	r_2	s_2	a_3	r_3	s_3	a_4	r_4	s_4	a_5	r_5	s_5
1	Bus	-1	1	Bus	-1	1	Bus	3	3	Walk	1	4	Walk	1	5

Q-Learning

Recall the Q-learning update:

$$\hat{Q}_{\text{opt}}(s,a) \leftarrow (1-\eta)\hat{Q}_{\text{opt}}(s,a) + \eta(r + \gamma \max_{a' \in \text{Action}(s')} \hat{Q}_{\text{opt}}(s',a'))$$

With $\eta = 0.5$ and $\gamma = 1$.

s_0	a_1	r_1	s_1	a_2	r_2	s_2	a_3	r_3	s_3	a_4	r_4	s_4	a_5	r_5	s_5
1	Bus	-1	1	Bus	-1	1	Bus	3	3	Walk	1	4	Walk	1	5

Find $\hat{Q}(s,a)$ for $s = 1,2,3,4$ and $a \in \{\text{Bus}, \text{Walk}\}$.

Q-Learning

s_0	a_1	r_1	s_1	a_2	r_2	s_2	a_3	r_3	s_3	a_4	r_4	s_4	a_5	r_5	s_5
1	Bus	-1	1	Bus	-1	1	Bus	3	3	Walk	1	4	Walk	1	5

$$\hat{Q}_{\text{opt}}(s,a) \leftarrow \frac{1}{2}\hat{Q}_{\text{opt}}(s,a) + \frac{1}{2}(r + \max_{a' \in \text{Action}(s')} \hat{Q}_{\text{opt}}(s',a'))$$

Using the updates:

$(1, \text{Bus}, -1, 1)$:

$(1, \text{Bus}, -1, 1)$:

$(1, \text{Bus}, 3, 3)$:

$(3, \text{Walk}, 1, 4)$:

$(4, \text{Walk}, 1, 5)$:

Q-Learning

s_0	a_1	r_1	s_1	a_2	r_2	s_2	a_3	r_3	s_3	a_4	r_4	s_4	a_5	r_5	s_5
1	Bus	-1	1	Bus	-1	1	Bus	3	3	Walk	1	4	Walk	1	5

$$\hat{Q}_{\text{opt}}(s,a) \leftarrow \frac{1}{2}\hat{Q}_{\text{opt}}(s,a) + \frac{1}{2}(r + \max_{a' \in \text{Action}(s')} \hat{Q}_{\text{opt}}(s',a'))$$

Using the updates:

$$(1, \text{Bus}, -1, 1) : \hat{Q}(1, \text{Bus}) = 0.5(0) + 0.5(-1 + 1\max(0,0)) = -0.5$$
$$(1, \text{Bus}, -1, 1) : \hat{Q}(1, \text{Bus}) = 0.5(-0.5) + 0.5(-1 + 1(\max(0, -0.5))) = -0.75$$
$$(1, \text{Bus}, 3, 3) : \hat{Q}(1, \text{Bus}) = 0.5(-0.75) + 0.5(3 + 1(\max(0,0))) = 1.125$$
$$(3, \text{Walk}, 1, 4) : \hat{Q}(3, \text{Walk}) = 0.5(0) + 0.5(1 + 1(\max(0,0))) = 0.5$$
$$(4, \text{Walk}, 1, 5) : \hat{Q}(4, \text{Walk}) = 0.5(0) + 0.5(1 + 1(\max(0,0))) = 0.5$$

All other $\hat{Q}(s,a) = 0$.

Algorithms for MDPs

Algorithms for MDPs

Model-Based Methods for MDPs

Unknown Transitions and Rewards: RL

Wouldn't it be nice if life gave you transition probabilities and rewards?

What if we only have:

- Starting state and possible states.
- Actions at each state.
- Termination state.
- Discount factor.

No transition probabilities and rewards.

Model-Based Value Iteration

As they say, when life gives you no transition or reward function, make your own!

Estimate the missing parts of the MDP (transition and rewards) by exploring the world and use value iteration to find the optimal policy. Hence 'model-based value iteration'.

$$\hat{T}(s,a,s') = \frac{|(s,a,s')|}{|(s,a)|} \quad \hat{R}(s,a,s') = r \text{ from } (s,a,r,s')$$

$$\hat{Q}_{\text{opt}}(s,a) = \sum_{s'} \hat{T}(s,a,s') \left[\hat{R}(s,a,s') + \gamma \hat{V}_{\text{opt}}(s')\right]$$

Value Iteration

$$\hat{Q}_{\text{opt}}(s,a) = \sum_{s'} \hat{T}(s,a,s') \left[\hat{R}(s,a,s') + \gamma \hat{V}_{\text{opt}}(s')\right]$$

$$V_{\text{opt}}(s) = \begin{cases} 0 & \text{if IsEnd}(s) \\ \max_{a \in A(s)} Q_{\text{opt}}(s,a) & \text{otherwise} \end{cases}$$

Algorithm: value iteration [Bellman, 1957]

Initialize $V_{\text{opt}}^{(0)}(s) \leftarrow 0$ for all states s.

For iteration $t = 1, \ldots, t_{\text{VI}}$:

For each state s:

$$V_{\text{opt}}^{(t)}(s) \leftarrow \max_{a \in \text{Actions}(s)} \underbrace{\sum_{s'} T(s,a,s')[\text{Reward}(s,a,s') + \gamma V_{\text{opt}}^{(t-1)}(s')]}_{Q_{\text{opt}}^{(t-1)}(s,a)}$$

Model-Based Monte Carlo

How do we explore the space?

Can't miss certain parts of the model (that deterministic π might). Solution? Make sure we randomly explore the space, visiting states infinitely often (in the limit).

Hence 'Model-Based Monte Carlo', randomly traverse the space.

Aside: On-Policy vs Off-Policy

Definition

On-Policy: estimate the value based on data generated by a specific policy.

Definition

Off-Policy: estimate the optimal value based on data generated by traversing the space.

Model-Based Monte Carlo? Off-Policy. We explore the space arbitrarily to find the optimal policy.

Algorithms for MDPs

Model-Free Methods for MDPs

Model-Free Monte Carlo

In *Model-Based Monte Carlo* we estimated $\hat{T}$ and $\hat{R}$ and used value iteration to compute $\hat{Q}_{\text{opt}}$. What if we just directly estimate $\hat{Q}_\pi(s, a)$?

Model-Free Monte Carlo

$\hat{Q}_\pi(s_{t-1}, a_t)$ should be the average of u_t, with:

$$u_t = r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \cdots$$

So at (s_{t-1}, a_t) use the rest of the data to get u_t.

Model-Free Monte Carlo

Equivalent formulation (convex combination)

On each (s, a, u):

$$\eta = \frac{1}{1+(\#\text{ updates to } (s,a))}$$

$$\hat{Q}_\pi(s,a) \leftarrow (1-\eta)\hat{Q}_\pi(s,a) + \eta u$$

Equivalent formulation (stochastic gradient)

On each (s, a, u):

$$\hat{Q}_\pi(s,a) \leftarrow \hat{Q}_\pi(s,a) - \eta[\underbrace{\hat{Q}_\pi(s,a)}_{\text{prediction}} - \underbrace{u}_{\text{target}}]$$

Implied objective: least squares regression

$$(\hat{Q}_\pi(s,a) - u)^2$$

SARSA

Notice that reaching state-action pair (s_{t-1}, a_t) required $u_t = \sum_{i=0}^{\infty} \gamma^i r_{t+i}$, which is the sum until termination, just for a single update. What if we updated every time we were at (s, a)?

SARSA For each tuple (s, a, r, s', a') in the sequence of our exploration (via π):

$$\hat{Q}_\pi(s, a) \leftarrow (1 - \eta)\hat{Q}_\pi(s, a) + \eta[r + \gamma \hat{Q}_\pi(s', a')]$$

Interpolate between observed data r and prediction.
Biased (using the estimate of $\hat{Q}_\pi$ rather than just raw data), but less variance.

SARSA vs Q-Learning

Algorithm: SARSA

On each (s, a, r, s', a'):

$$\hat{Q}_\pi(s,a) \leftarrow (1-\eta)\hat{Q}_\pi(s,a) + \eta[\underbrace{r}_{\text{data}} + \gamma \underbrace{\hat{Q}_\pi(s',a')}_{\text{estimate}}]$$

Algorithm: Q-learning [Watkins/Dayan, 1992]

On each (s, a, r, s'):

$$\hat{Q}_{\text{opt}}(s,a) \leftarrow (1-\eta)\underbrace{\hat{Q}_{\text{opt}}(s,a)}_{\text{prediction}} + \eta\underbrace{(r + \gamma \hat{V}_{\text{opt}}(s'))}_{\text{target}}$$

Recall: $\hat{V}_{\text{opt}}(s') = \max_{a' \in \text{Actions}(s')} \hat{Q}_{\text{opt}}(s', a')$

Exploration/Exploitation Trade-off

In Q-Learning we need some policy to generate data while we estimate another policy (the optimal one). Does any policy work?

- Too greedy (always picking the best action) and we won't explore everywhere.
- Too much exploring and we learn too slowly.

Solution? ϵ-greedy policy:

$$\pi_{\text{act}}(s) = \begin{cases} \text{argmax}_{a \in A(s)} \hat{Q}_{\text{opt}}(s, a) & \text{probability } 1 - \epsilon \\ \text{random action from } A(s) & \text{probability } \epsilon \end{cases}$$

Can decay ϵ over time, guarantees we explore and learn.

Algorithms for MDPs

Summary

Reinforce Your Understanding

Off-policy algorithms that output the optimal Q-value, Q_{opt}:

- **Value Iteration**: $V_{\text{opt}}^{(t)}(s) \leftarrow \max_{a \in A(s)} Q_{\text{opt}}^{(t-1)}(s, a)$.
- **Model-Based Value Iteration**: Estimate T and R using Monte Carlo, then value iteration using estimates $\hat{T}$ and $\hat{R}$.
- **Q-Learning**: Estimate $\hat{Q}_{\text{opt}}(s, a)$ based on (i) the reward to state s' and (ii) the estimated optimal max value of s'.

On-policy algorithms that output the Q-value, Q_π, of a specific policy:

- **Policy Iteration**: $V_\pi^{(t)}(s) \leftarrow Q_\pi^{(t-1)}(s, \pi(s))$.
- **Model-Free Monte Carlo**: Estimate $\hat{Q}_\pi(s, a)$ from the utility, u_t, along the path.
- **SARSA**: Estimate $\hat{Q}_\pi(s, a)$ based on (i) the update (s, a, r, s', a') and (ii) the estimated $\hat{Q}_\pi(s', a')$.

Reinforce Your Understanding

Algorithm	Outputs	Based On
Model-Based Monte Carlo	$\hat{Q}_{\text{opt}}$	$s_0, a_1, r_1, s_1, \ldots \implies \hat{T}, \hat{R}$
Q-Learning	$\hat{Q}_{\text{opt}}$	$(s, a, r, s'), \hat{V}_{\text{opt}}(s')$
Model-Free Monte Carlo	$\hat{Q}_{\pi}$	$u_t = r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \ldots$
SARSA	$\hat{Q}_{\pi}$	$(s, a, r, s', a'), \hat{Q}_{\pi}(s', a')$

Problem 2: Choosing an Algorithm

Problem 2: Choosing an Algorithm

Choosing an Algorithm

For each of the following questions, choose one or more of the algorithm(s) below to solve the given problem.

- Model-Based Monte Carlo
- Q-Learning
- Model-Free Monte Carlo
- SARSA

Expensive Experiments

You work in a chemistry lab that is conducting some *extremely* expensive experiments. Unfortunately, sometimes the actions you take cause non-deterministic outcomes (due to unobservable factors), and your chemical reaction transitions to a different state randomly. Your team would like to figure out the best course of action that'll most likely finish the experiments without fail, but you don't have enough budget for lots of trials. Fortunately, the number of states and possible actions are relatively small, and you have detailed notes on data from many past experiments.

Expensive Experiments

You work in a chemistry lab that is conducting some *extremely* expensive experiments. Unfortunately, sometimes the actions you take cause non-deterministic outcomes (due to unobservable factors), and your chemical reaction transitions to a different state randomly. Your team would like to figure out the best course of action that'll most likely finish the experiments without fail, but you don't have enough budget for lots of trials. Fortunately, the number of states and possible actions are relatively small, and you have detailed notes on data from many past experiments.

Model-Based Monte Carlo would let us simulate many strategies (policies) using old and new data by generating $\hat{T}$ and $\hat{R}$ before using Policy Iteration.

Poker-Playing Roommates

Your roommate seems to win a lot of money playing poker against their friends. Based on what you know about them, you've got some ideas on what strategies you can use to possibly beat them. But, you're concerned that if you tailor your strategy to your roommate's specific playstyle, then you'll lose to their friends in the crossfire. You decide to build a poker bot to test your strategies against random players online. You choose your reward to be how much you win at the end of a round.

Poker-Playing Roommates

Your roommate seems to win a lot of money playing poker against their friends. Based on what you know about them, you've got some ideas on what strategies you can use to possibly beat them. But, you're concerned that if you tailor your strategy to your roommate's specific playstyle, then you'll lose to their friends in the crossfire. You decide to build a poker bot to test your strategies against random players online. You choose your reward to be how much you win at the end of a round.

Model-Free Monte Carlo is a good choice since we only get a reward at the end of a round (at a terminal state), meaning that SARSA updates would be slower. Since we are testing specific policies, we need an on-policy algorithm.

Monte Catlo?

You decide to foster a cat from the local Humane Society. Unfortunately, the cat is quite skittish and really likes the dark. As such, he won't get out from under your bed. You've assembled an arsenal of treats, toys, and trinkets to try and lure him out. Some things seem to pique his interest, but he won't seem to come out. You're pretty sure that presenting him with the right order of items at the right time of day might convince him to come out.

Monte Catlo?

You decide to foster a cat from the local Humane Society. Unfortunately, the cat is quite skittish and really likes the dark. As such, he won't get out from under your bed. You've assembled an arsenal of treats, toys, and trinkets to try and lure him out. Some things seem to pique his interest, but he won't seem to come out. You're pretty sure that presenting him with the right order of items at the right time of day might convince him to come out.

Q-Learning would work well here, since we'd like to learn an optimal policy (get the cat out from under the bed), and have no idea how he reacts to things. Alternatively, Model-Based Monte Carlo.