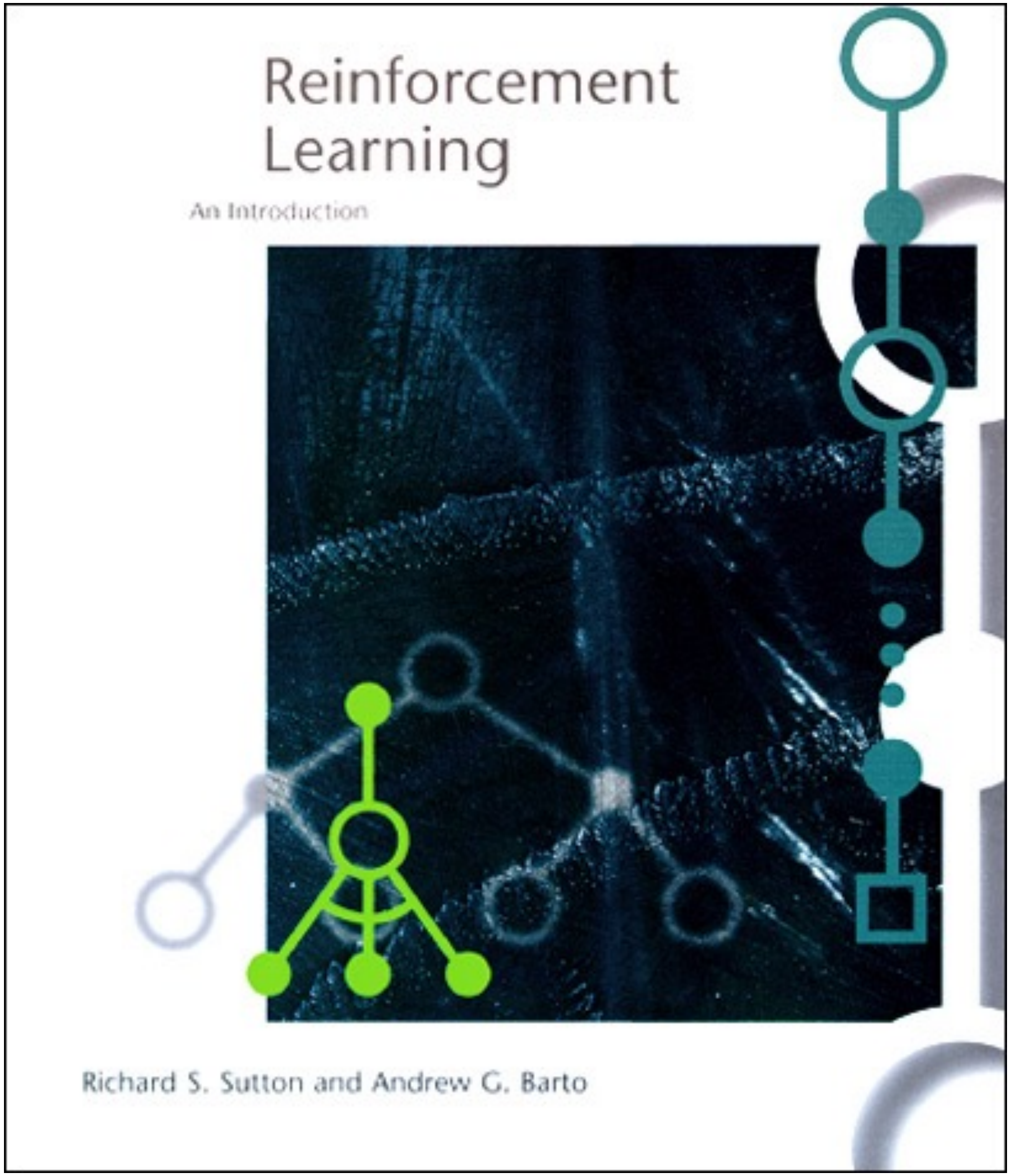
CMSC 471: Reinforcement Learning

Some slides courtesy Cynthia Matuszek and Frank Farrero, with some material from Marie desJardin, Lise Getoor, Jean-Claude Latombe, and Daphne Koller

There’s an entire book!

http://incompleteideas.net/book/the-book-2nd.html



The Big Idea

- "Planning": Find a sequence of steps to accomplish a goal.
  - Given start state, transition model, goal functions…
- This is a kind of **sequential decision making**.
  - Transitions are deterministic.
- What if they are stochastic (probabilistic)?
  - One time in ten, you drop your sock
- Probabilistic Planning: Make a plan that accounts for probability by **carrying it through the plan**.

Review: Formalizing Agents

- Given:
  - A state space S
  - A set of actions $a_1, \ldots, a_k$ including their results
  - Reward value at the end of each trial (series of action) (may be positive or negative)
- Output:
  - A **mapping from states to actions**

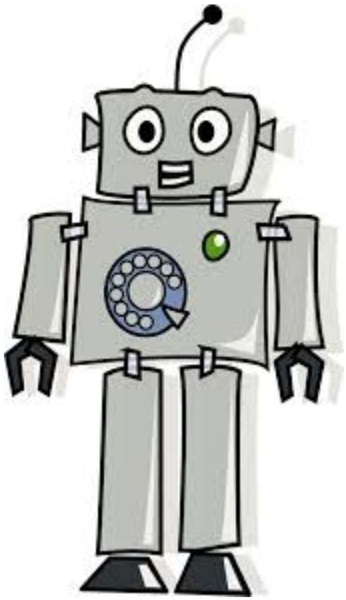
Review: Formalizing Agents

- Given:
  - A state space S
  - A set of actions $a_1, \ldots, a_k$ including their results
  - Reward value at the end of each trial (series of action) (may be positive or negative)
- Output:
  - A **mapping from states to actions**
  - Which is a **policy**, π

Reinforcement Learning

- We often have an agent which has a **task** to perform
  - It takes some actions in the world
  - At some later point, gets feedback on how well it did
  - The agent performs the same task repeatedly
- This problem is called **reinforcement learning**:
  - The agent gets positive reinforcement for tasks done well
  - And gets negative reinforcement for tasks done poorly
  - Must somehow figure out which actions to take next time

Reinforcement Learning



agent

environment

Robot image: openclipart.org

https://static.vecteezy.com/system/resources/previews/000/090/451/original/four-seasons-landscape-illustrations-vector.jpg

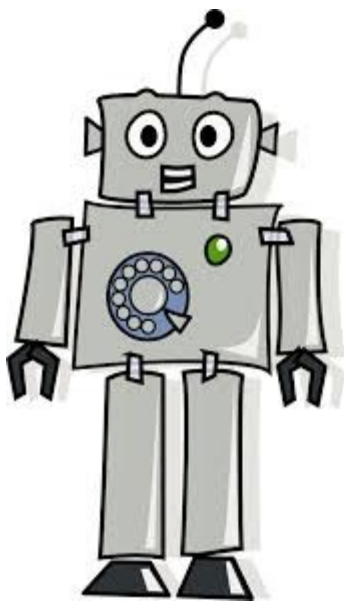
Reinforcement Learning

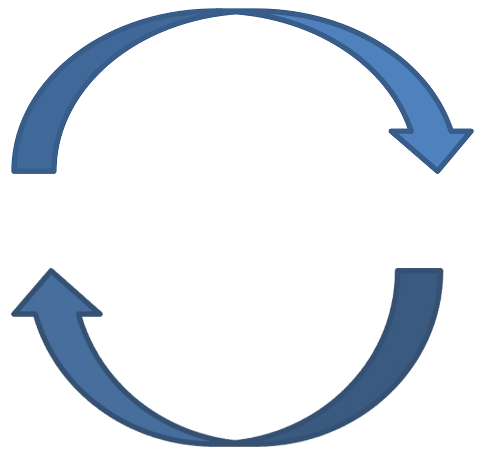
Robot image: openclipart.org

https://static.vecteezy.com/system/resources/previews/000/090/451/original/four-seasons-landscape-illustrations-vector.jpg

Reinforcement Learning

take action





get new state
and/or reward

environment

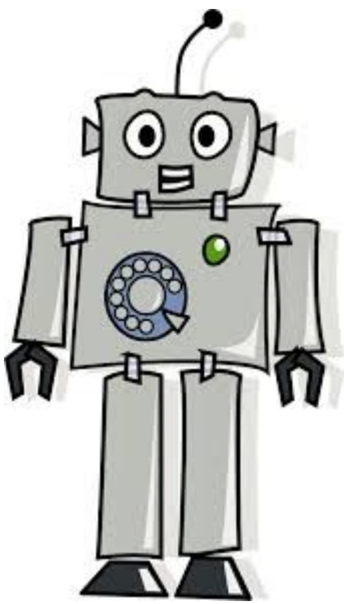
agent

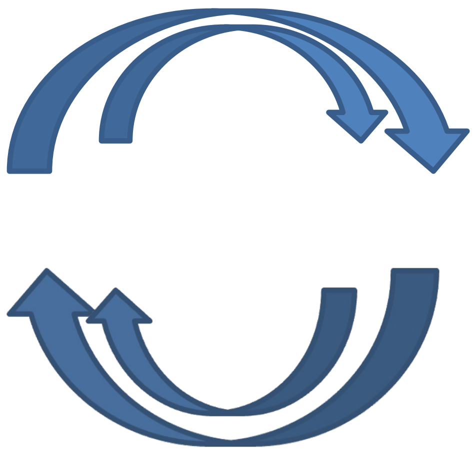
Robot image: openclipart.org

https://static.vecteezy.com/system/resources/previews/000/090/451/original/four-seasons-landscape-illustrations-vector.jpg

Reinforcement Learning

take action

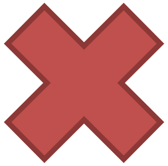




get new state
and/or reward

environment

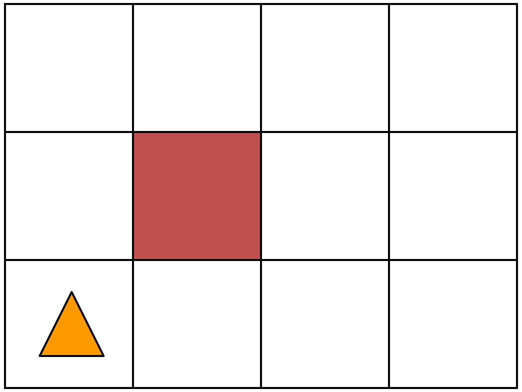
agent



Robot image: openclipart.org

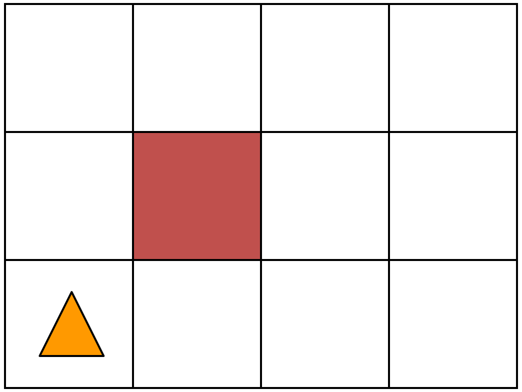
https://static.vecteezy.com/system/resources/previews/000/090/451/original/four-seasons-landscape-illustrations-vector.jpg

Simple Robot Navigation Problem



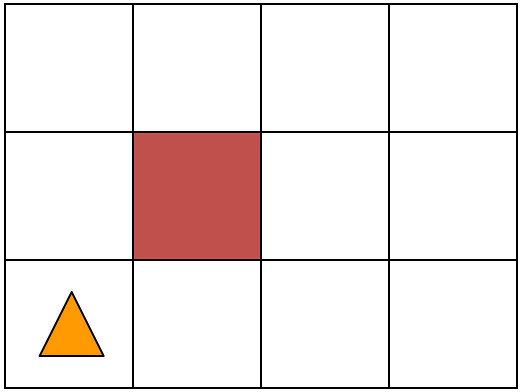
- In each state, the possible actions are U, D, R, and L

Probabilistic Transition Model



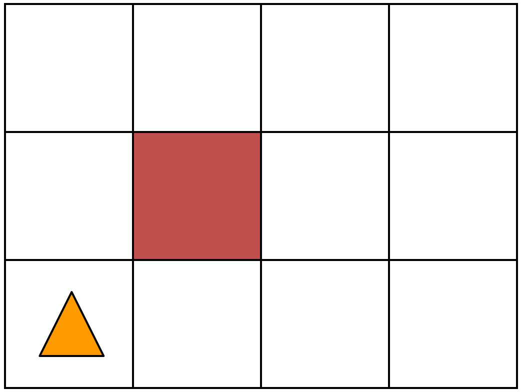
- In each state, the possible actions are U, D, R, and L
- The effect of U is as follows (transition model):
    - With probability 0.8, the robot moves up one square (if the robot is already in the top row, then it does not move)

Probabilistic Transition Model



- In each state, the possible actions are U, D, R, and L
- The effect of U is as follows (transition model):
  - With probability 0.8, the robot moves up one square (if the robot is already in the top row, then it does not move)
  - With probability 0.1, the robot moves right one square (if the robot is already in the rightmost row, then it does not move)

Probabilistic Transition Model



- In each state, the possible actions are U, D, R, and L
- The effect of U is as follows (transition model):
  - With probability 0.8, the robot moves up one square (if the robot is already in the top row, then it does not move)
  - With probability 0.1, the robot moves right one square (if the robot is already in the rightmost row, then it does not move)
  - With probability 0.1, the robot moves left one square (if the robot is already in the leftmost row, then it does not move)

Markov Property

The transition properties depend only on the current state, not on the previous history (how that state was reached)

Markov assumption generally: current state only ever depends on previous state (or finite set of previous states).

Markov Decision Process: Formalizing Reinforcement Learning

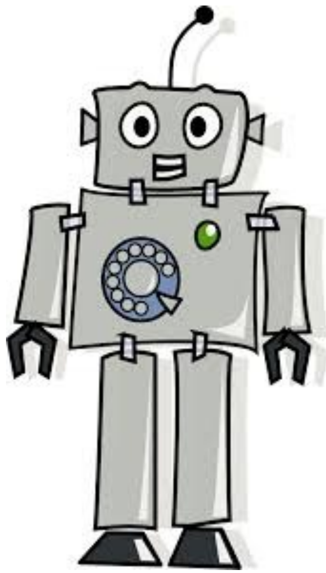
Markov Decision Process: $(\mathcal{S}, \mathcal{A}, \mathcal{R}, P, \gamma)$

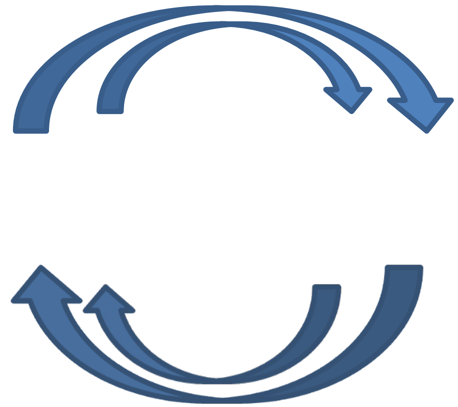
Robot image: openclipart.org

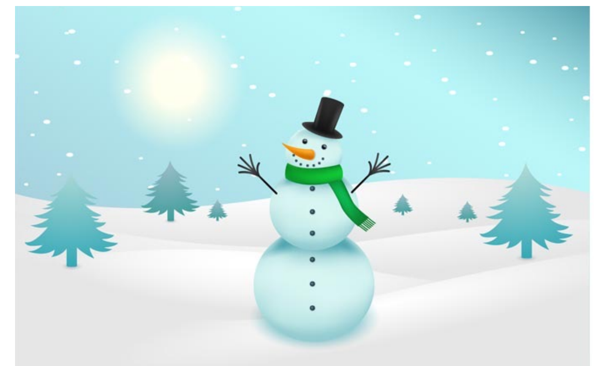
https://static.vecteezy.com/system/resources/previews/000/090/451/original/four-seasons-landscape-illustrations-vector.jpg

Markov Decision Process: Formalizing Reinforcement Learning

take action







get new state and/or reward

environment

agent

set of possible actions

Markov Decision Process: $(\mathcal{S}, \mathcal{A}, \mathcal{R}, P, \gamma)$

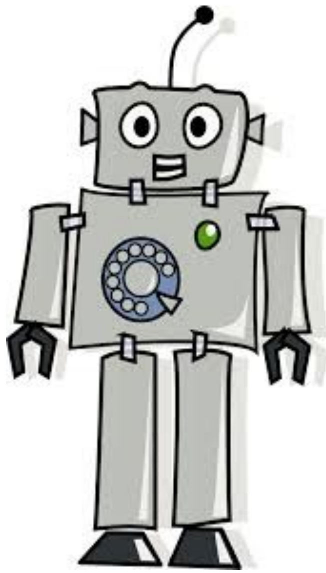
set of possible states

Robot image: openclipart.org

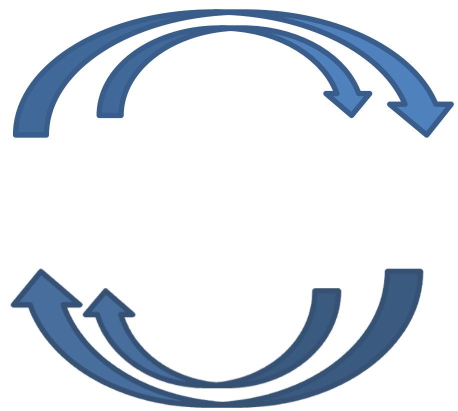
https://static.vecteezy.com/system/resources/previews/000/090/451/original/four-seasons-landscape-illustrations-vector.jpg

Markov Decision Process: Formalizing Reinforcement Learning

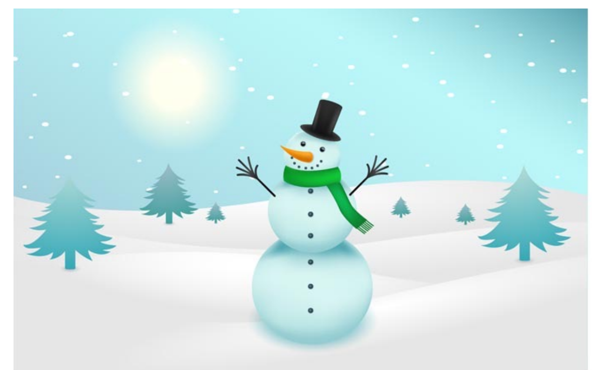
take action



agent



get new state
and/or reward



environment

Markov Decision Process:

set of possible actions

$$(\mathcal{S}, \mathcal{A}, \mathcal{R}, P, \gamma)$$

set of possible states

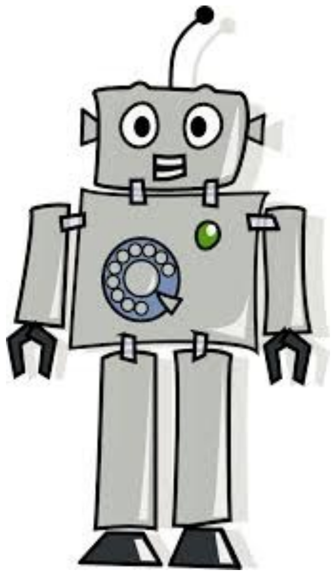
reward of (state, action) pairs

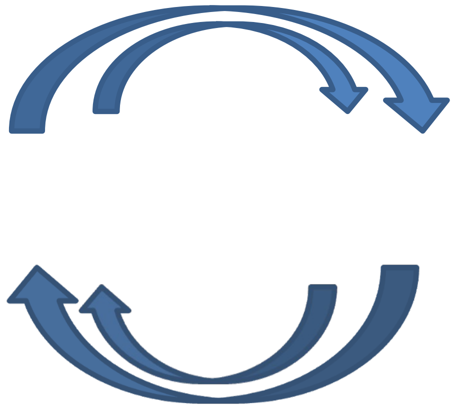
Robot image: openclipart.org

https://static.vecteezy.com/system/resources/previews/000/090/451/original/four-seasons-landscape-illustrations-vector.jpg

Markov Decision Process: Formalizing Reinforcement Learning

take action





get new state and/or reward

environment

agent

Markov Decision Process:

set of possible actions

state-action transition distribution

$$(\mathcal{S}, \mathcal{A}, \mathcal{R}, P, \gamma)$$

set of possible states

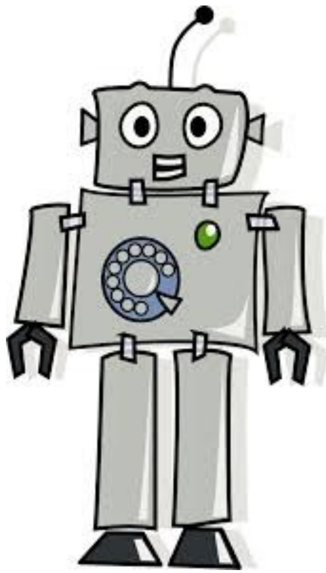
reward of (state, action) pairs

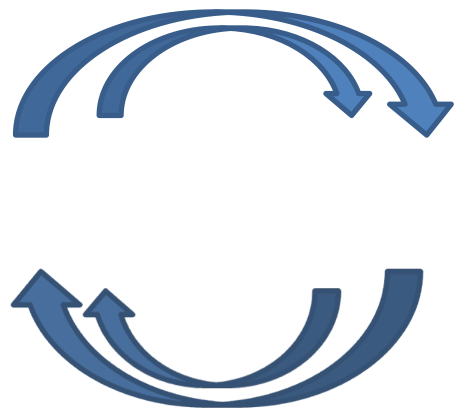
Robot image: openclipart.org

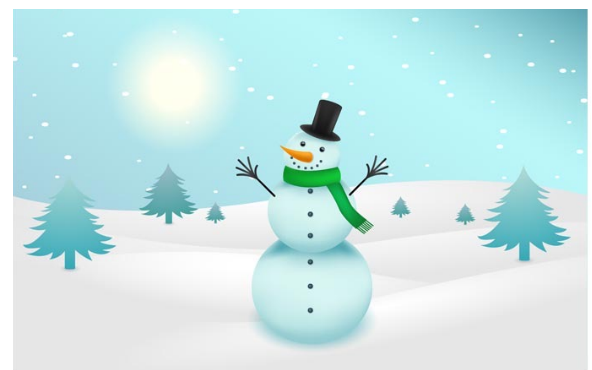
https://static.vecteezy.com/system/resources/previews/000/090/451/original/four-seasons-landscape-illustrations-vector.jpg

Markov Decision Process: Formalizing Reinforcement Learning

take action







get new state
and/or reward

environment

agent

set of possible actions

state-action transition distribution

Markov Decision Process:

$$(\mathcal{S}, \mathcal{A}, \mathcal{R}, P, \gamma)$$

set of possible states

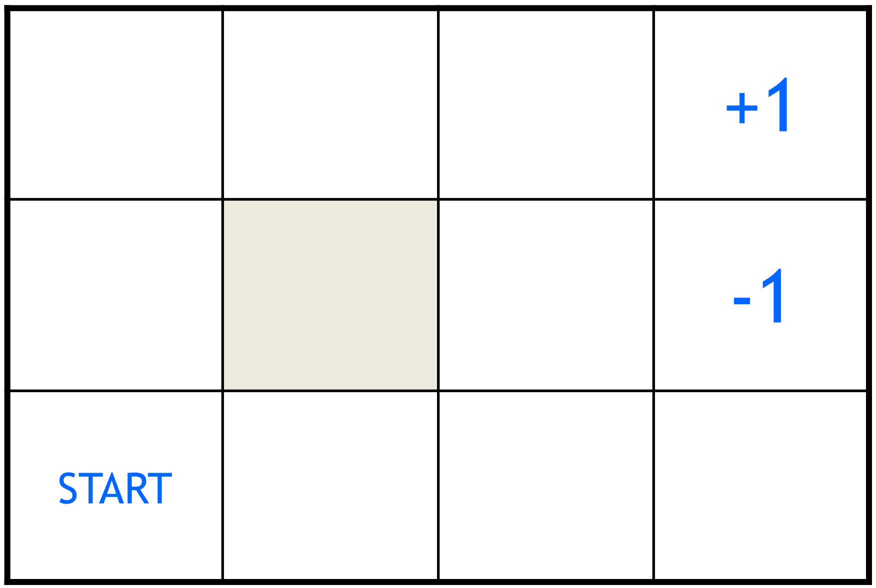
reward of (state, action) pairs

discount factor

Robot image: openclipart.org

https://static.vecteezy.com/system/resources/previews/000/090/451/original/four-seasons-landscape-illustrations-vector.jpg

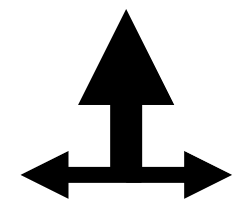
Robot in a room



actions: UP, DOWN, LEFT, RIGHT

UP

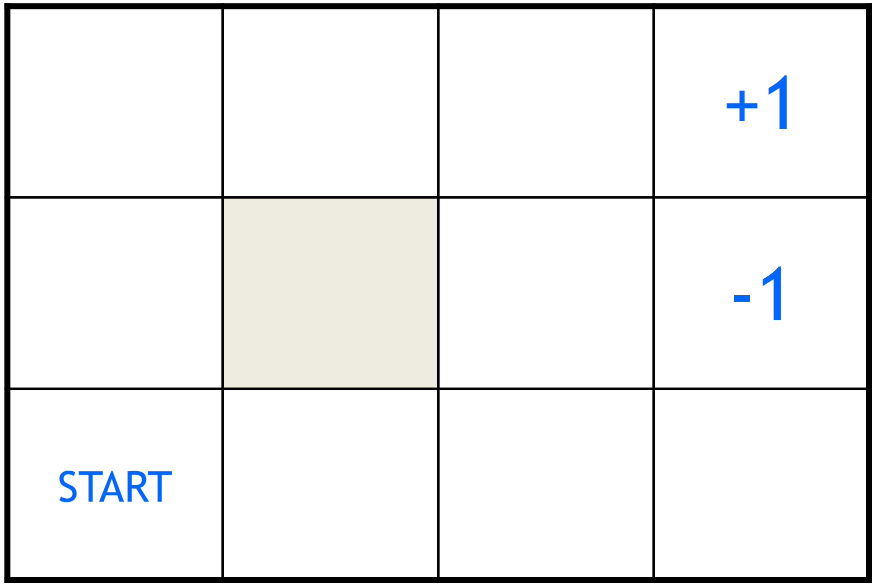
80% move UP
10% move LEFT
10% move RIGHT



reward +1 at [4,3], -1 at [4,2]
reward -0.04 for each step

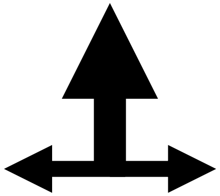
Goal: what's the strategy to achieve the maximum reward?

Robot in a room



actions: UP, DOWN, LEFT, RIGHT

UP

80%	move UP	
10%	move LEFT	
10%	move RIGHT	

reward +1 at [4,3], -1 at [4,2]
reward -0.04 for each step

states: current location
actions: where to go next
rewards

what is the solution? Learn a mapping from (state, action) pairs to new states

Slide courtesy Peter Bodík

Markov Decision Process: Formalizing Reinforcement Learning

Markov Decision Process:

$$(\mathcal{S}, \mathcal{A}, \mathcal{R}, P, \gamma)$$

set of possible states

set of possible actions

reward of (state, action) pairs

state-action transition distribution

discount factor

Start in initial state s_0

Markov Decision Process: Formalizing Reinforcement Learning

Markov Decision Process:

$$\underbrace{(\mathcal{S}}_{\text{set of possible states}}, \overbrace{\mathcal{A}}^{\text{set of possible actions}}, \underbrace{\mathcal{R}}_{\text{reward of (state, action) pairs}}, \overbrace{P}^{\text{state-action transition distribution}}, \underbrace{\gamma}_{\text{discount factor}})$$

Start in initial state s_0
for t = 1 to …:
  choose action a_t

Markov Decision Process: Formalizing Reinforcement Learning

set of possible actions

state-action transition distribution

Markov Decision Process: $(\mathcal{S}, \mathcal{A}, \mathcal{R}, P, \gamma)$

set of possible states

reward of (state, action) pairs

discount factor

Start in initial state s_0
for t = 1 to …:
    choose action a_t
    "move" to next state $s_t \sim \pi(\cdot | s_{t-1}, a_t)$

$Policy$
π: S → A

Markov Decision Process: Formalizing Reinforcement Learning

Markov Decision Process: $(\mathcal{S}, \mathcal{A}, \mathcal{R}, P, \gamma)$

- $\mathcal{S}$: set of possible states
- $\mathcal{A}$: set of possible actions
- $\mathcal{R}$: reward of (state, action) pairs
- P: state-action transition distribution
- γ: discount factor

Start in initial state s_0
for t = 1 to …:
  choose action a_t
  “move” to next state $s_t \sim \pi(\cdot|s_{t-1}, a_t)$
  get reward $r_t = \mathcal{R}(s_t, a_t)$

Markov Decision Process: Formalizing Reinforcement Learning

set of possible actions

state-action transition distribution

Markov Decision Process: $(\mathcal{S}, \mathcal{A}, \mathcal{R}, P, \gamma)$

set of possible states

reward of (state, action) pairs

discount factor

Start in initial state s_0
for t = 1 to …:
  choose action a_t
  "move" to next state $s_t \sim \pi(\cdot | s_{t-1}, a_t)$
  get reward $r_t = \mathcal{R}(s_t, a_t)$

objective: choose action over time to maximize time-discounted reward

Markov Decision Process: Formalizing Reinforcement Learning

set of possible actions

state-action transition distribution

Markov Decision Process: $(\mathcal{S}, \mathcal{A}, \mathcal{R}, P, \gamma)$

set of possible states

reward of (state, action) pairs

discount factor

Start in initial state s_0
for t = 1 to …:
  choose action a_t
  "move" to next state $s_t \sim \pi(\cdot | s_{t-1}, a_t)$
  get reward $r_t = \mathcal{R}(s_t, a_t)$

objective: choose action over time to maximize discounted reward

Consider all possible future times t

Reward at time t

Markov Decision Process: Formalizing Reinforcement Learning

Markov Decision Process:

set of possible actions

state-action transition distribution

$$(\mathcal{S}, \mathcal{A}, \mathcal{R}, P, \gamma)$$

set of possible states

reward of (state, action) pairs

discount factor

Start in initial state s_0
for t = 1 to …:
  choose action a_t
  "move" to next state $s_t \sim \pi(\cdot|s_{t-1}, a_t)$
  get reward $r_t = \mathcal{R}(s_t, a_t)$

objective: maximize discounted reward

Consider all possible future times t

Discount at time t

Reward at time t

Markov Decision Process: Formalizing Reinforcement Learning

Markov Decision Process: $(\mathcal{S}, \mathcal{A}, \mathcal{R}, P, \gamma)$

- $\mathcal{S}$: set of possible states
- $\mathcal{A}$: set of possible actions
- $\mathcal{R}$: reward of (state, action) pairs
- P: state-action transition distribution
- γ: discount factor

Start in initial state s_0
for t = 1 to …:
  choose action a_t
  "move" to next state $s_t \sim \pi(\cdot|s_{t-1}, a_t)$
  get reward $r_t = \mathcal{R}(s_t, a_t)$

objective: maximize discounted reward

$$\max_{\pi} \sum_{t>0} \gamma^t r_t$$

Consider all possible future times t

Discount at time t

Reward at time t

Example of Discounted Reward

objective: maximize discounted reward

$$\max_{\pi} \sum_{t>0} \gamma^t r_t$$

Consider all possible future times t

Discount at time t

Reward at time t

- If the discount factor $\gamma = 0.8$ then reward
$$0.8^0 r_0 + 0.8^1 r_1 + 0.8^2 r_2 + 0.8^3 r_3 + \cdots + 0.8^n r_n + \ \ldots$$
- Allows you to consider all possible rewards in the future but preferring current vs. future self

Markov Decision Process: Formalizing Reinforcement Learning

Markov Decision Process: $(\mathcal{S}, \mathcal{A}, \mathcal{R}, P, \gamma)$

- $\mathcal{S}$: set of possible states
- $\mathcal{A}$: set of possible actions
- $\mathcal{R}$: reward of (state, action) pairs
- P: state-action transition distribution
- γ: discount factor

Start in initial state s_0
for t = 1 to …:
  choose action a_t
  "move" to next state $s_t \sim \pi(\cdot | s_{t-1}, a_t)$
  get reward $r_t = \mathcal{R}(s_t, a_t)$

objective: maximize discounted reward

$$\max_{\pi} \sum_{t>0} \gamma^t r_t$$

"solution": the policy π^* that maximizes the expected (average) time-discounted reward

Markov Decision Process: Formalizing Reinforcement Learning

Markov Decision Process: $(\mathcal{S}, \mathcal{A}, \mathcal{R}, P, \gamma)$

- $\mathcal{S}$: set of possible states
- $\mathcal{A}$: set of possible actions
- $\mathcal{R}$: reward of (state, action) pairs
- P: state-action transition distribution
- γ: discount factor

Start in initial state s_0
for t = 1 to …:
  choose action a_t
  "move" to next state $s_t \sim \pi(\cdot | s_{t-1}, a_t)$
  get reward $r_t = \mathcal{R}(s_t, a_t)$

objective: maximize discounted reward

$$\max_{\pi} \sum_{t>0} \gamma^t r_t$$

"solution" $$\pi^* = \operatorname*{argmax}_{\pi} \mathbb{E}\left[\sum_{t>0} \gamma^t r_t \, ; \pi\right]$$