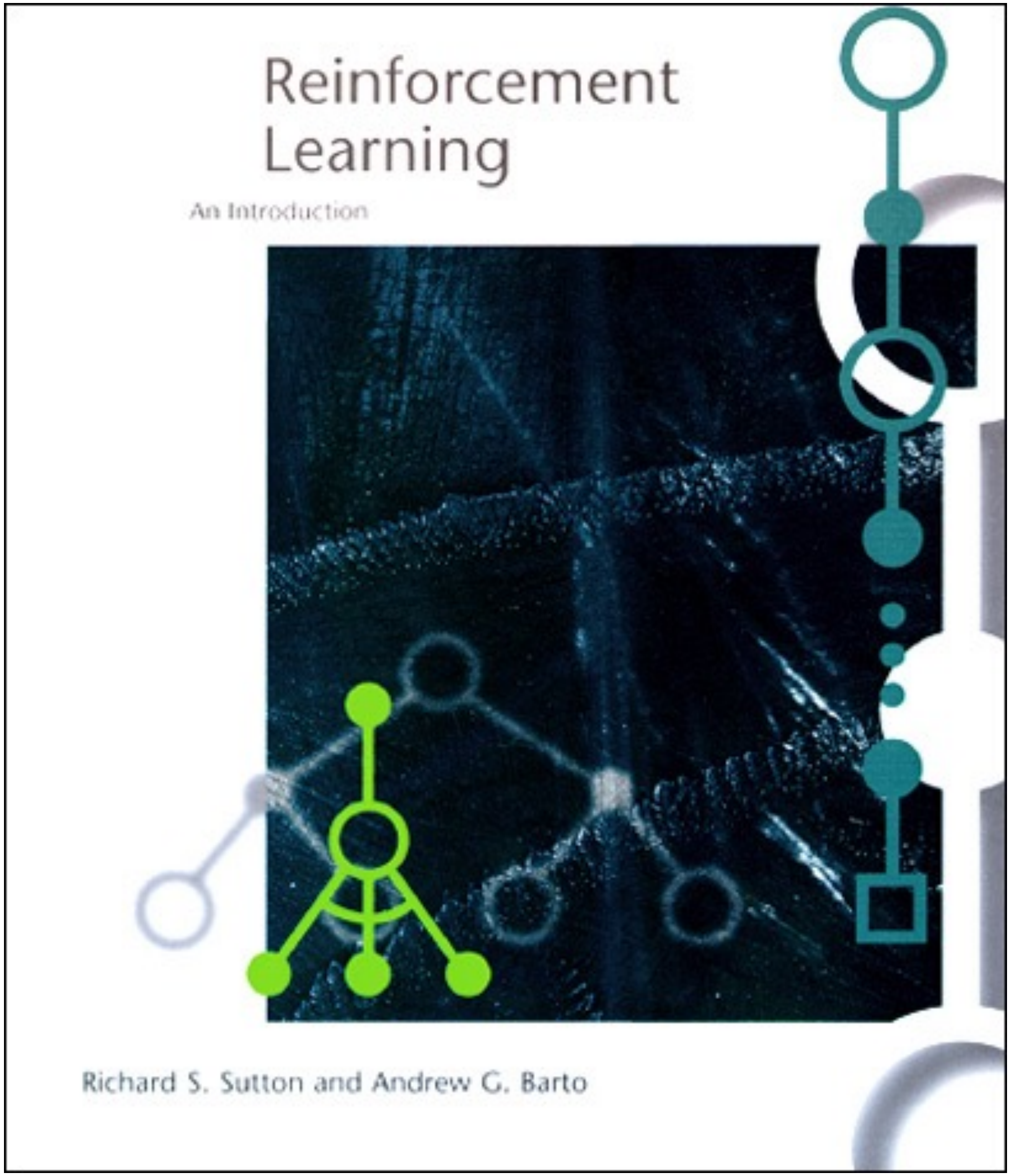
CMSC 478: Reinforcement Learning

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, …, 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**

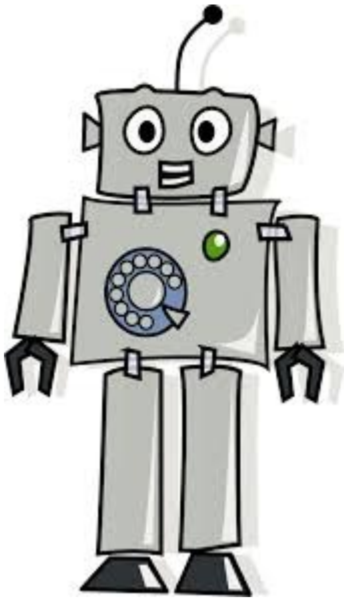
Review: Formalizing Agents

- 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

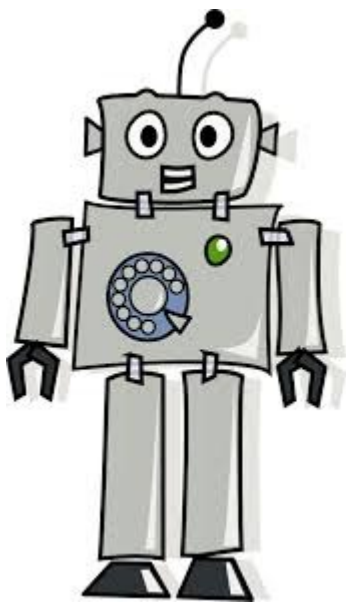
environment

Robot image: openclipart.org

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

Reinforcement Learning

take action

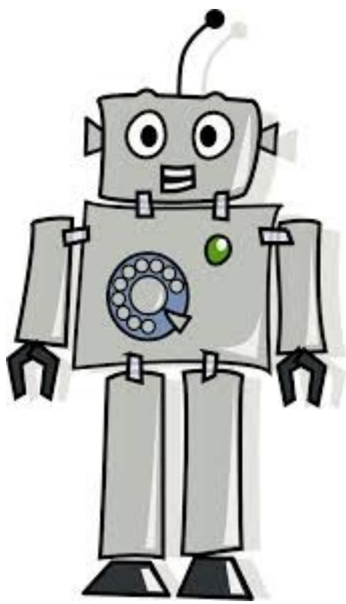


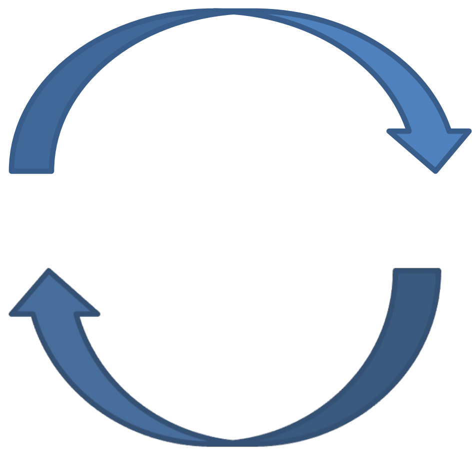
agent

environment

Reinforcement Learning

take action





get new state
and/or reward

environment

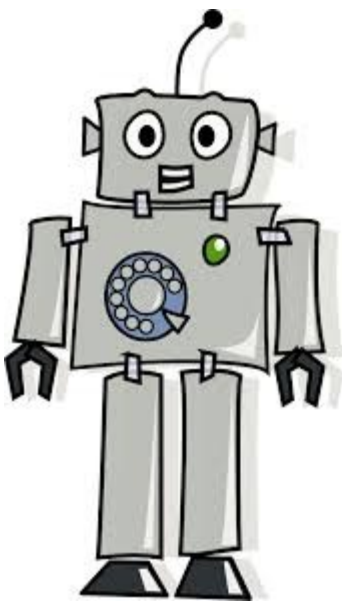
agent

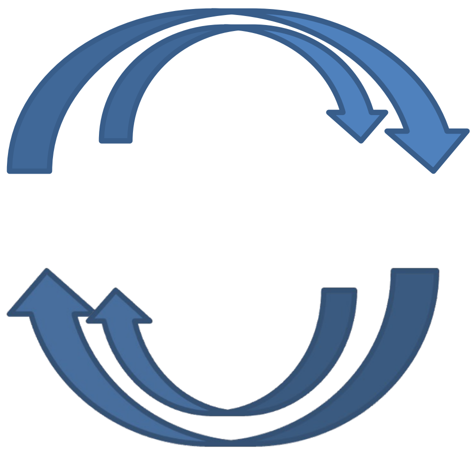
Robot image: openclipart.org

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

Reinforcement Learning

take action





agent

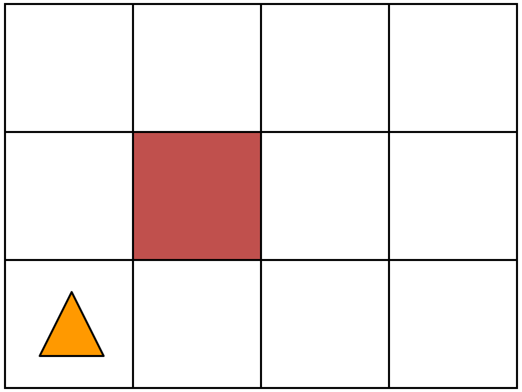
get new state
and/or reward

environment

Robot image: openclipart.org

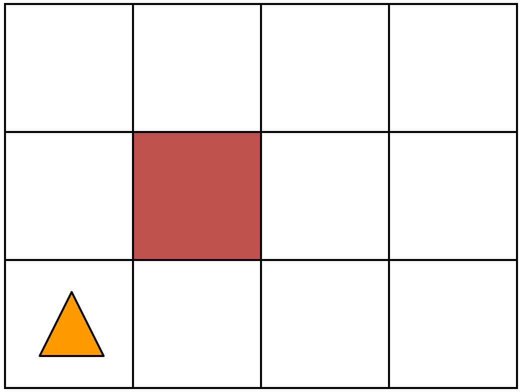
https://static.vecteezy.com/system/resources/previews/000/090/451/original/four-seasons-landscape-illustrations-vector.jpg

Simple Robot Navigation Problem



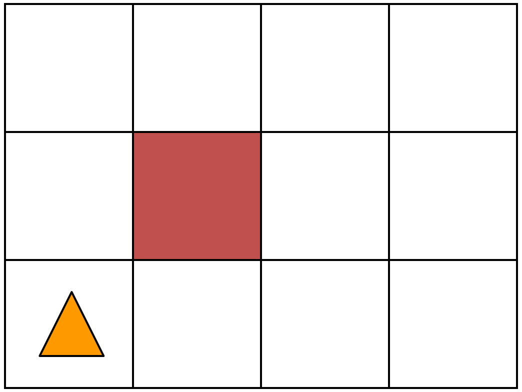
- In each state, the possible actions are U, D, R, and L

Probabilistic Transition Model



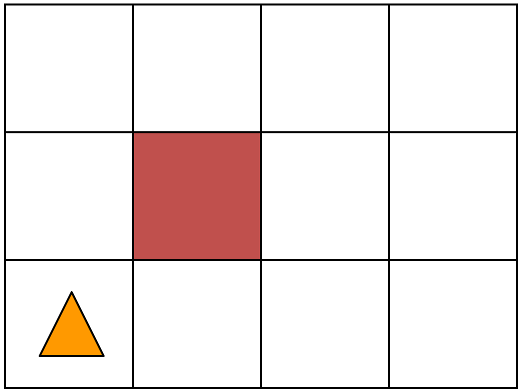
- In each state, the possible actions are U, D, R, and L
- The effect of U is as follows (transition model):
  - With probability 0.8, the robot moves up one square (if the robot is already in the top row, then it does not move)

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

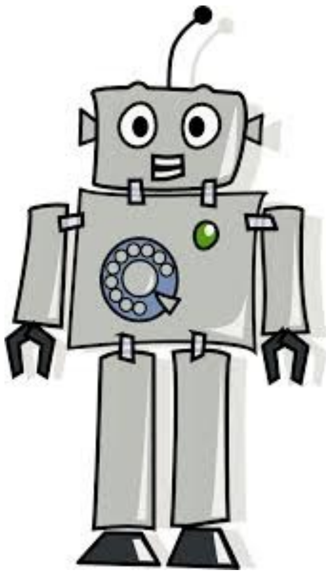
Markov Decision Process: $(\mathcal{S}, \mathcal{A}, \mathcal{R}, P, \gamma)$

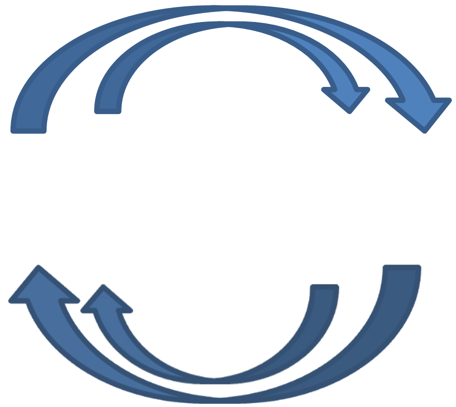
Robot image: openclipart.org

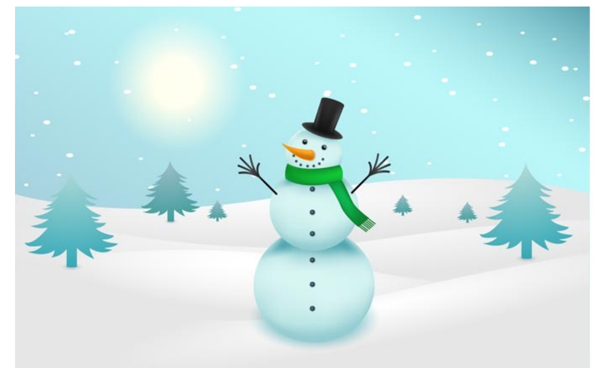
https://static.vecteezy.com/system/resources/previews/000/090/451/original/four-seasons-landscape-illustrations-vector.jpg

Markov Decision Process: Formalizing Reinforcement Learning

take action







get new state
and/or reward

environment

agent

set of
possible
actions

Markov Decision Process:

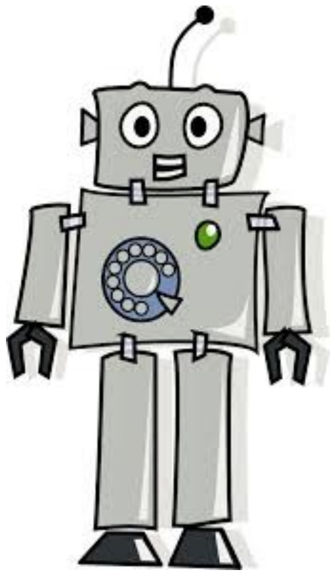
$$(\mathcal{S}, \mathcal{A}, \mathcal{R}, P, \gamma)$$

set of
possible
states

Robot image: openclipart.org

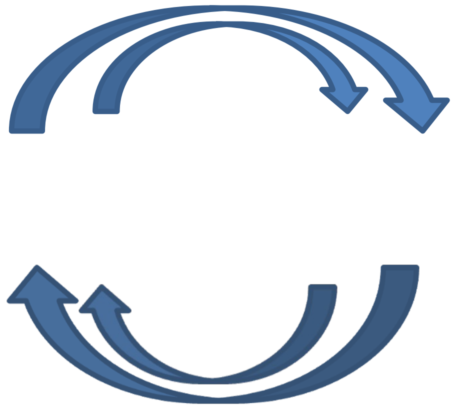
https://static.vecteezy.com/system/resources/previews/000/090/451/original/four-seasons-landscape-illustrations-vector.jpg

Markov Decision Process: Formalizing Reinforcement Learning



agent

take action



get new state and/or reward

environment

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

$\mathcal{S}$: set of possible states

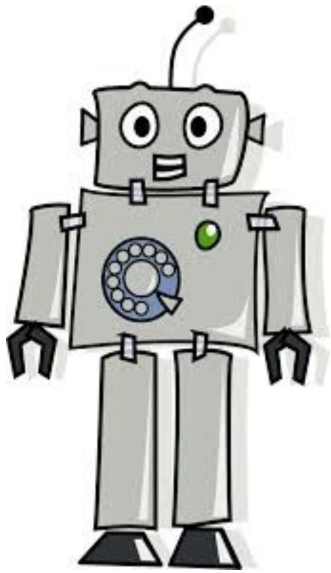
$\mathcal{A}$: set of possible actions

$\mathcal{R}$: reward of (state, action) pairs

Robot image: openclipart.org

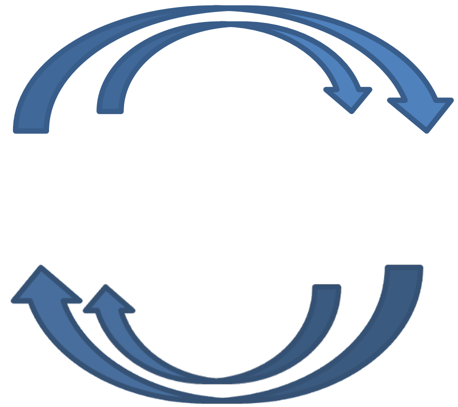
https://static.vecteezy.com/system/resources/previews/000/090/451/original/four-seasons-landscape-illustrations-vector.jpg

Markov Decision Process: Formalizing Reinforcement Learning



agent

take action



get new state and/or reward

environment

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

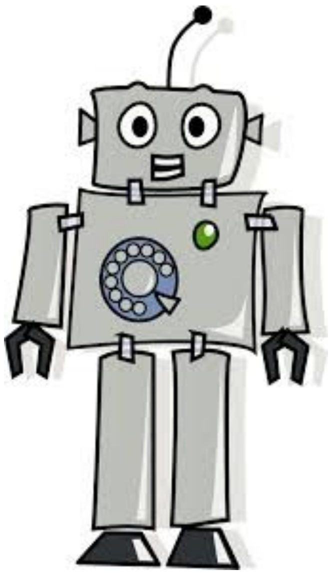
P: state-action transition distribution

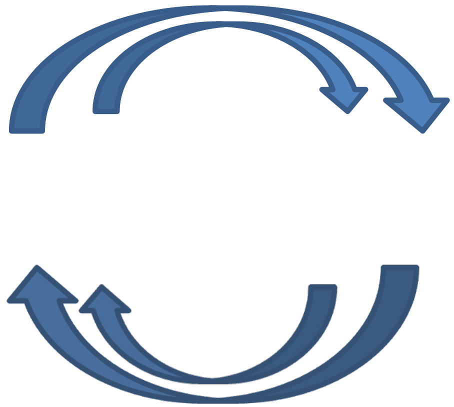
Robot image: openclipart.org

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:

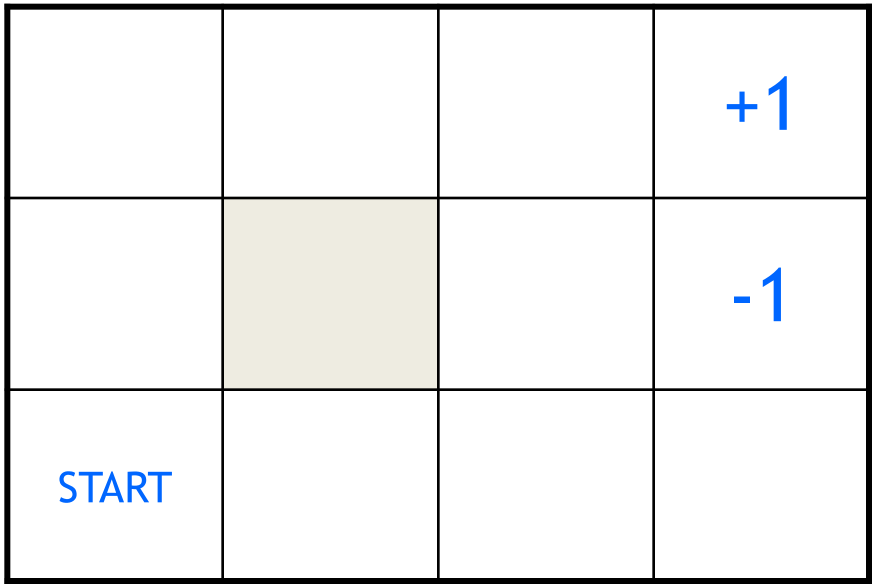
$$(\mathcal{S}, \mathcal{A}, \mathcal{R}, P, \gamma)$$

- $\mathcal{S}$: 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

Robot image: openclipart.org

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

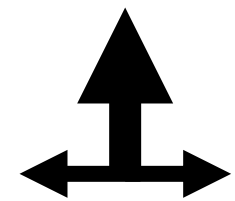
Robot in a room



actions: UP, DOWN, LEFT, RIGHT

UP

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



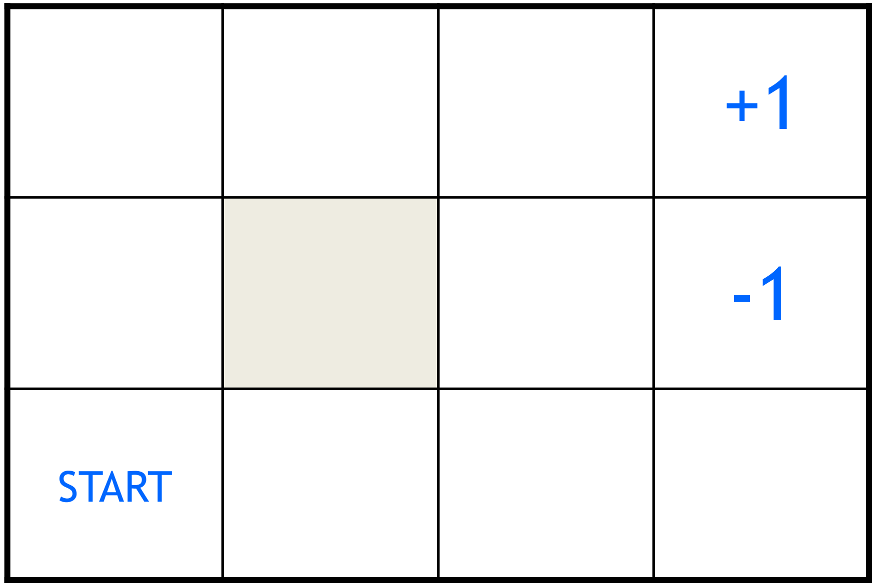
reward +1 at [4,3], -1 at [4,2]

reward -0.04 for each step

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

Slide courtesy Peter Bodík

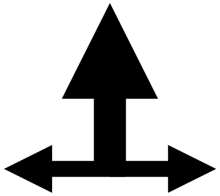
Robot in a room



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

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

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

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 $s_t \sim \pi(\cdot|s_{t-1}, a_t)$

$Policy$
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

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: 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

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

set of possible states ($\mathcal{S}$)

set of possible actions ($\mathcal{A}$)

reward of (state, action) pairs ($\mathcal{R}$)

state-action transition distribution (P)

discount factor (γ)

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

Markov Decision Process: Formalizing Reinforcement Learning

Mar

Here, r_t is a function of random variable s_t.

Start in initia

for t = 1 to ..

choose act

"move" to next state $s_t \sim \pi(\cdot | s_{t-1}, a_t)$

get reward $r_t = \mathcal{R}(s_t, a_t)$

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

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

Markov Decision Process: Formalizing Reinforcement Learning

Mar

Here, r_t is a function of random variable s_t. ➔
The expectation is over the different states s_t the agent could be in at time *t* (equiv. actions the agent could take).

Start in initia

for t = 1 to ..

choose act a_t

"move" to next state $s_t \sim \pi(\cdot|s_{t-1}, a_t)$

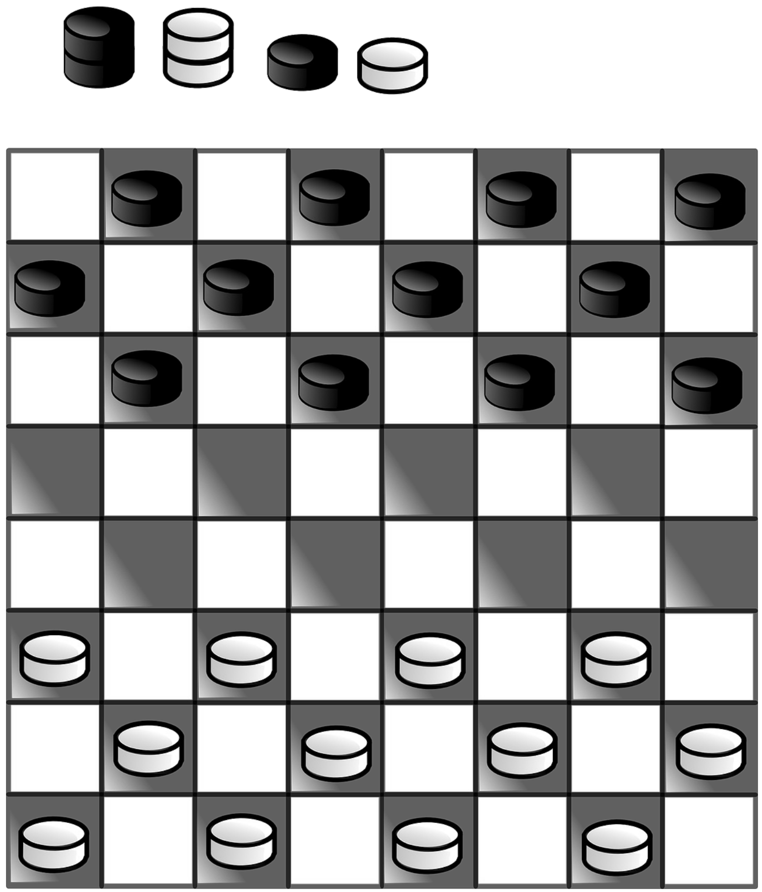
get reward $r_t = \mathcal{R}(s_t, a_t)$

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

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

Simple Example

- Learn to play checkers
  - Two-person game
  - 8x8 boards, 12 checkers/side
  - relatively simple set of rules: http://www.darkfish.com/checkers/rules.html
  - Goal is to eliminate all your opponent’s pieces



https://pixabay.com/en/checker-board-black-game-pattern-29911

Some Challenges

1. Representing states (and actions)

2. Defining our reward

3. Learning our policy

Overview: Learning Strategies

Dynamic Programming

Q-learning

Monte Carlo approaches

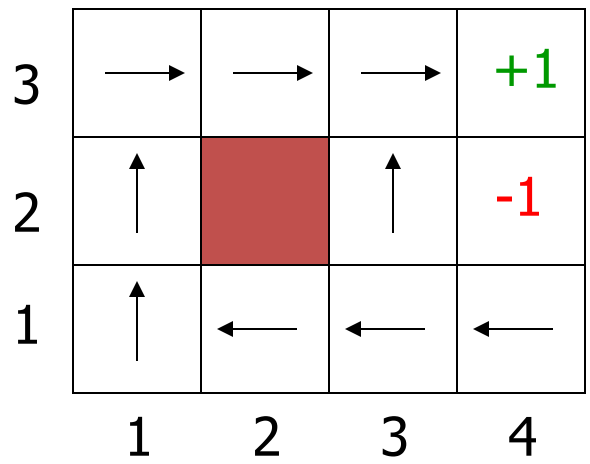
Reactive Agent Algorithm

Accessible or observable state

Repeat:

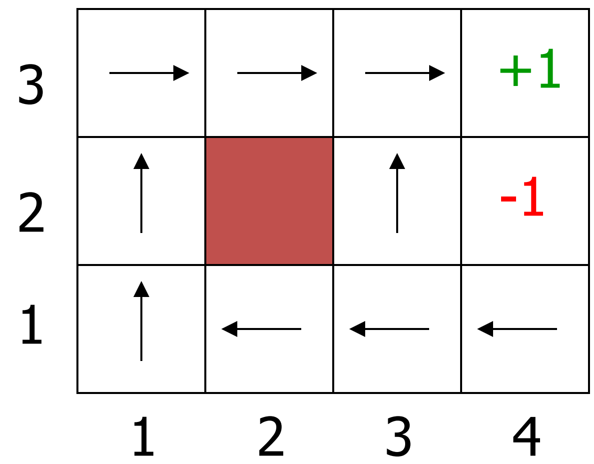
- s ← sensed state
- If s is a terminal state then exit
- a ← choose action (given s)
- Perform a

Policy (Reactive/Closed-Loop Strategy)



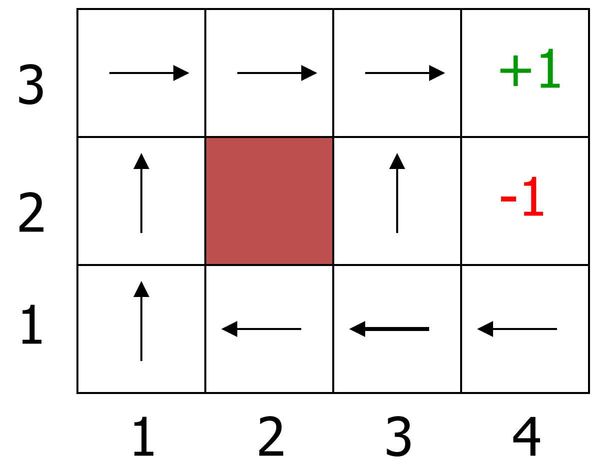
- In every state, we need to know what to do
- The **goal** doesn't change
- A policy (Π) is a complete mapping from *states* to *actions*
  - "If in [3,2], go up; if in [3,1], go left; if in…"

Optimal Policy



- A policy Π is a complete mapping from states to actions
- The optimal policy Π* is the one that always yields a history (sequence of steps ending at a terminal state) with maximal ***expected*** utility

Optimal Policy



- A policy Π is a comp ... ns
- The optimal policy Π ... history with maximal expected utility

This problem is called a Markov Decision Problem (MDP)

How to compute Π*?

Defining Value Function

- Problem:
  - When making a decision, we only know the reward so far, and the possible actions

Defining Value Function

- Problem:
  - When making a decision, we only know the reward so far, and the possible actions
  - We've defined value function retroactively (i.e., the value function/utility of a history/sequence of states is known *once we finish it*)

Defining Value Function

- Problem:
  - When making a decision, we only know the reward so far, and the possible actions
  - We've defined value function retroactively (i.e., the value function/utility of a history/sequence of states is known *once we finish it*)
  - What is the value function of a particular ***state*** in the middle of decision making?

Defining Value Function

- Problem:
  - When making a decision, we only know the reward so far, and the possible actions
  - We've defined value function retroactively (i.e., the value function/utility of a history/sequence of states is known *once we finish it*)
  - What is the value function of a particular ***state*** in the middle of decision making?
  - Need to compute ***expected value function*** of possible future histories/states

Defining Value Function

$$V^{\pi}(s) = \mathrm{E}\left[R(s_0) + \gamma R(s_1) + \gamma^2 R(s_2) + \cdots \,\middle|\, s_0 = s, \pi\right].$$

$V^{\pi}(s)$ is simply the expected sum of discounted rewards upon starting in state s, and taking actions according to π.[1]

Given a fixed policy π, its value function V^{π} satisfies the **Bellman equations**:

$$V^{\pi}(s) = R(s) + \gamma \sum_{s' \in S} P_{s\pi(s)}(s') V^{\pi}(s').$$

- What is the value function of a particular ***state*** in the middle of decision making?
- Need to compute ***expected value function*** of possible future histories/states

Dynamic programming

use value functions to structure the search for good policies

Dynamic programming

use value functions to structure the search for good policies

policy evaluation: compute V^{π} from π

policy improvement: improve π based on V^{π}

Slide courtesy/adapted: Peter Bodík

Dynamic programming

use value functions to structure the search for good policies

policy evaluation: compute V^{π} from π

policy improvement: improve π based on V^{π}

start with an arbitrary policy

repeat evaluation/improvement until convergence