DEPARTAMENTO DE INFORMÁTICA
PUC-RIO

MDP: Markov Decision Problem

Deep Reinforcement Learning

Alberto Sardinha
sardinha@inf.puc-rio.br

Outline

- **Markov decision problem**
- Optimal state-value function and optimal action-value function
- Value iteration
- Example

Markov Decision Problem (MDP)

- A framework for **sequential decision making** of **a single agent**

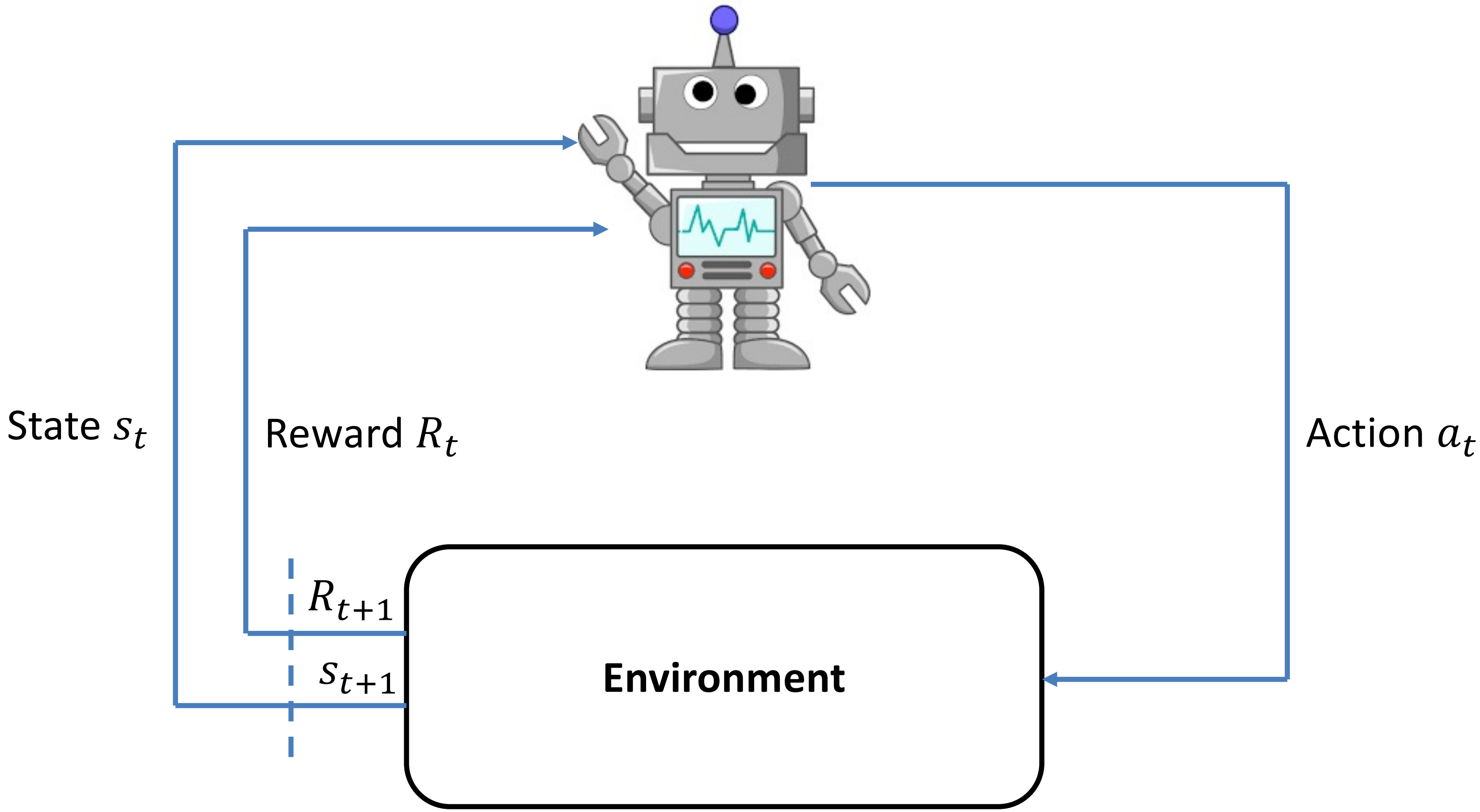
- **Markovian transition model** and **fully observable**
  - i.e., we assume it verifies the Markov property and $s_t = \theta_t$

- **Planning horizon** can be infinite

Markov Decision Problem (MDP)

- We can formally define an MDP with following elements:
  - **Discrete time** $t = 0, 1, 2, \ldots$
  - A **discrete set of states** $s \in S$
  - A **discrete set of actions** $a \in A$
  - A **stochastic transition model** $P(s'|s, a)$
    - the world transitions stochastically to state s' when the agent takes action a at state s
  - A **reward function** $R: S \times A \rightarrow \mathbb{R}$
    - An agent receives a reward $R(s, a)$ when it takes action a at state s
  - A **discount rate** γ

Markov Decision Problem (MDP)



Markov Decision Problem (MDP)

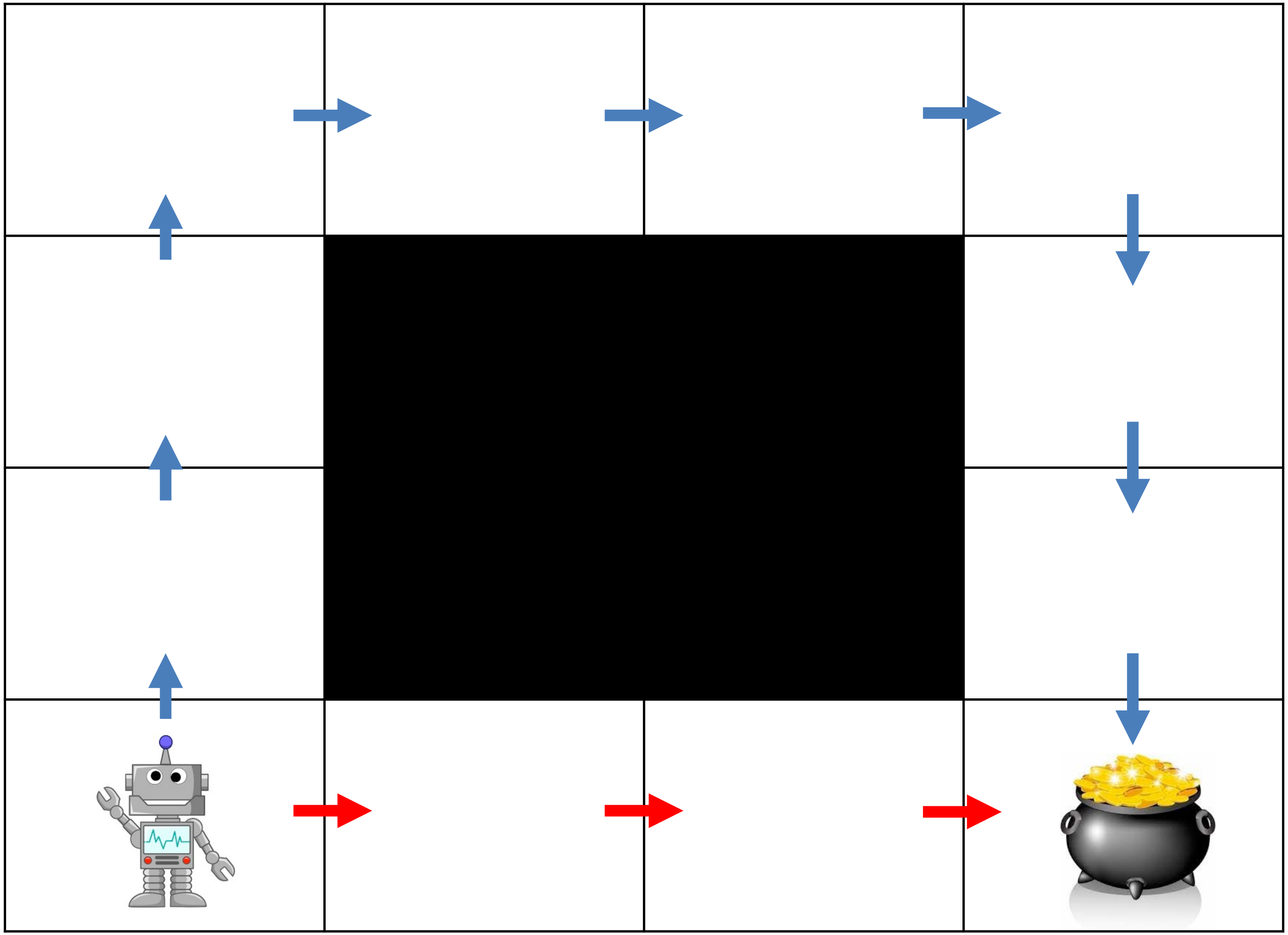
- The **task** of the agent is to **maximize a function of accumulated reward** over its planning horizon

- A standard function is the **discounted future reward**:

$$R(s_t, a_t) + \gamma R(s_{t+1}, a_{t+1}) + \gamma^2 R(s_{t+2}, a_{t+2}) + \ \ldots$$

- where γ is the **discount rate** and $0 \leq \gamma \leq 1$
  - γ is a value between 0 and 1 in order to ensure the sum remains finite for infinite horizon

Which trajectory would you choose?



What is the best sequence of actions in this MDP?

- Discounted future reward for the blue trajectory:

$$0 + \gamma\, 0 + \gamma^2\, 0 + \gamma^3\, 0 + \gamma^4\, 0 + \gamma^5\, 0 + \gamma^6\, 0 + \gamma^7\, 0 + \gamma^8\, 100$$

- Discounted future reward for the red trajectory:

$$0 + \gamma\, 0 + \gamma^2\, 100$$

- The red sequence has the highest value!

Markov Decision Problem (MDP)

- A **stationary policy** $\pi : S \to A$ of the agent in an MDP is a mapping $\pi(s)$ from states to actions.

$$\pi : S \to A$$

- **Different policies** will produce **different discounted future rewards**
  - Each policy will take the agent through different trajectories

Markov Decision Problem (MDP)

- **Definition**: the **state-value function** of a state s under a policy π is the expected return the agent can receive when starting in state s and then following policy π :

$$V^{\pi}(s) = E\left[\sum_{t=0}^{\infty} \gamma^{t} R(s_t, a_t) | s_0 = s, a_t = \pi(s_t)\right]$$

Definition: the **action-value function (Q-values)** of taking an action a in state s under a policy π is the expected return the agent can receive when starting in state s, taking action a, and then following policy π :

$$Q^{\pi}(s, a) = E\left[\sum_{t=0}^{\infty} \gamma^{t} R(s_t, a_t) | s_0 = s, a_0 = a, a_{t>0} = \pi(s_t)\right]$$

Outline

- Markov decision problem
- **Optimal state-value function and optimal action-value function**
- Value iteration
- Example

Markov Decision Problem (MDP)

- A **policy π is defined to be better than or equal to a policy π'** if the expected return of π is greater or equal to expected return of π':

$$\pi \geq \pi' \text{ if and only if } V^{\pi}(s) \geq V^{\pi'}(s), \text{ for all } s \in S$$

- **There is always at least one policy** that is better than or equal to all other policies, which is the **optimal policy**
  - Note that an MDP might have more than one optimal policy
  - We denote all the optimal polices by π^*

Markov Decision Problem (MDP)

- These policies π^* share the same state-value function, called **optimal state-value function**, with the following definition:

$$V^*(s) = \max_{\pi} V^{\pi}(s), \text{ for all } s \in S$$

- These policies π^* also share the same action-value function, called **optimal action-value function**, with the following definition:

$$Q^*(s, a) = \max_{\pi} Q^{\pi}(s, a), \text{ for all } s \in S \text{ and } a \in A$$

Markov Decision Problem (MDP)

- **But how do we find this optimal policy?**
  - Naïve solution: compute the state-value function for all policies

Outline

- Markov decision problem
- Optimal state-value function and optimal action-value function
- **Value iteration**
- Example

Markov Decision Problem (MDP)

- Lets us now define the **stationary policy** π of the agent in an MDP as a mapping from states to a distribution over actions

$$\pi: S \rightarrow \Delta(A)$$

- An example of a distribution for an MDP with $S = \{x, y\}$, $A = \{b, c\}, P, R$:

$$\pi = \begin{array}{cc} & \begin{array}{cc} b \quad & \quad c \end{array} \\ \begin{array}{c} x \\ y \end{array} & \begin{bmatrix} 1 & 0 \\ 0.5 & 0.5 \end{bmatrix} \end{array}$$

- Thus, we can use the notation $\pi(a|s)$
  - For instance, $\pi(a = b|s = x) = 1$

Markov Decision Problem (MDP)

$$V^{\pi}(s) = E\left[\sum_{t=0}^{\infty} \gamma^t R(s_t\,, a_t) | s_0 = s, a_t = \pi(s_t)\right]$$

$$V^{\pi}(s) = E_{\pi}\left[\sum_{t=0}^{\infty} \gamma^t R_t \;| s_0 = s\right]$$

$$V^{\pi}(s) = E_{\pi}[R_0 + \gamma R_1 + \gamma^2 R_2 + \cdots | s_0 = s]$$

Markov Decision Problem (MDP)

$$V^{\pi}(s) = E_{\pi}[R_0 + \gamma R_1 + \gamma^2 R_2 + \cdots |s_0 = s]$$

$$V^{\pi}(s) = E_{\pi}[R_0 + \gamma (R_1 + \gamma R_2 + \cdots)|s_0 = s]$$

$$V^{\pi}(s) = E_{\pi}[R_0|s_0 = s] + E_{\pi}[\gamma (R_1 + \gamma R_2 + \cdots)|s_0 = s]$$

$$V^{\pi}(s) = R_{\pi}(s) + E_{\pi}[\gamma (R_1 + \gamma R_2 + \cdots)|s_0 = s]$$

$$\boxed{R_{\pi}(s) = E_{\pi}[R(s,a)] = \sum_{a \epsilon A} \pi(a|s) R(s,a)}$$

Policy-averaged reward

Markov Decision Problem (MDP)

We would like this to be s_1

$$V^{\pi}(s) = R_{\pi}(s) + \gamma E_{\pi}[(R_1 + \gamma R_2 + \cdots)|s_0 = s]$$

We use the law of total probability with expectation

$$V^{\pi}(s) = R_{\pi}(s) + \gamma \sum_{s' \epsilon S} E_{\pi}[(R_1 + \gamma R_2 + \cdots)|s_1 = s'] P_{\pi}(s'|s)$$

$$\boxed{\begin{aligned} P_{\pi}(s'|s) &= E_{\pi}[P(s'|s,a)] \\ &= \sum_{a \in A} \pi(a|s) P(s'|s,a) \end{aligned}}$$

Policy-averaged probabilities

$$\boxed{V^{\pi}(s) = R_{\pi}(s) + \gamma \sum_{s' \epsilon S} P_{\pi}(s'|s)\, V^{\pi}(s')}$$

Markov Decision Problem (MDP)

- We now take recursion

$$V^{\pi}(s) = R_{\pi}(s) + \gamma \sum_{s' \epsilon S} P_{\pi}(s'|s)\, V^{\pi}(s')$$

- And make the following substitutions

$$V^{\pi}(s) = \sum_{a \epsilon A} \pi(a|s) R(s,a) + \gamma \sum_{s' \epsilon S} \sum_{a \in A} \pi(a|s) P(s'|s,a)\, V^{\pi}(s')$$

$$\boxed{V^{\pi}(s) = \sum_{a \epsilon A} \pi(a|s) \Big[\underbrace{R(s,a) + \gamma \sum_{s' \epsilon S} P(s'|s,a)\, V^{\pi}(s')}_{Q^{\pi}(s,a)} \Big]}$$

Markov Decision Problem (MDP)

- The **optimal state-value function** can be written in a special form without reference to a specific policy (so-called **Bellman equation**):

$$V^*(s) = \max_{a \in A} \left[R(s,a) + \gamma \sum_{s' \epsilon S} P(s'|s,a) V^*(s') \right]$$

- A similar recursive equation holds for the **Q-values**:

$$Q^*(s,a) = R(s,a) + \gamma \sum_{s' \in S} P(s'|s,a) \max_{a' \in A} Q^*(s',a')$$

Value Iteration

- **Value iteration** is a **simple and efficient method for computing optimal values in an MDP**

Value Iteration

- We initialize arbitrarily a state-value function (e.g., with zeros, ones, etc.)

- Then we iteratively apply the Bellman equation (in the previous slides) turned into an assignment operation:

$$Q(s,a) := R(s,a) + \gamma \sum_{s' \epsilon S} P(s'|s,a)V(s'),\ \forall s, \forall a$$

$$V(s) := \max_{a \in A} Q(s,a), \forall s$$

- Repeat the above two equations until V does not change significantly between two consecutive steps

Value Iteration

- Value iteration converges to the optimal Q^* for any initialization

- After computing the optimal Q^*, we can extract the policy as follows:

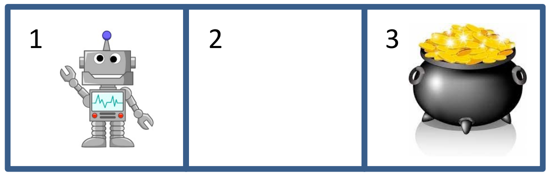
$$\pi^*(s) \in \underset{a \in A}{\operatorname{argmax}}\, Q^*(s, a)$$

Outline

- Markov decision problem
- Optimal state-value function and optimal action-value function
- Value iteration
- **Example**

Example

- We have the following MDP:
  - $S = \{1, 2, 3\}$
  - $A = \{left, right\}$
  - $P(s'|s, a = left) = \begin{bmatrix} 1 & 0 & 0 \\ 0.8 & 0.2 & 0 \\ 0 & 0.8 & 0.2 \end{bmatrix}$
  - $P(s'|s, a = right) = \begin{bmatrix} 0.2 & 0.8 & 0 \\ 0 & 0.2 & 0.8 \\ 0 & 0 & 1 \end{bmatrix}$

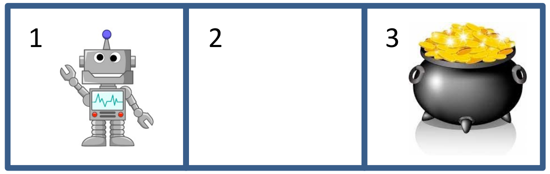


Example

- We have the following MDP:

  - $R(s,a) = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 1 & 1 \end{bmatrix}$

  - $\gamma = 0.9$



Example

- Let us use value iteration:
  - We initialize state-value function with ones

$$V = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$$

Example

- we iteratively apply the following equation ($i = 0$)

$$Q(s,a) \coloneqq R(s,a) + \gamma \sum_{s' \epsilon S} P(s'|s,a)V(s'), \ \forall s, \forall a$$

$$Q(1, left) \coloneqq 0 + 0.9(1\times1 + 0\times1 + 0\times1) = 0.9$$
$$Q(2, left) \coloneqq 0 + 0.9(0.8\times1 + 0.2\times1 + 0\times1) = 0.9$$
$$Q(3, left) \coloneqq 1 + 0.9(0\times1 + 0.8\times1 + 0.2\times1) = 1.9$$

$$Q(1, right) \coloneqq 0 + 0.9(0.2\times1 + 0.8\times1 + 0\times1) = 0.9$$
$$Q(2, right) \coloneqq 0 + 0.9(0\times1 + 0.2\times1 + 0.8\times1) = 0.9$$
$$Q(3, right) \coloneqq 1 + 0.9(0\times1 + 0\times1 + 1\times1) = 1.9$$

Example

- we iteratively apply the following equations ($i = 0$)

$$Q(s,a) := R(s,a) + \gamma \sum_{s' \epsilon S} P(s'|s,a)V(s'),\ \ \forall s, \forall a$$

$$Q = \begin{bmatrix} 0.9 & 0.9 \\ 0.9 & 0.9 \\ 1.9 & 1.9 \end{bmatrix}$$

$$V(s) := \max_{a \in A} Q(s,a), \forall s$$

$$V = \begin{bmatrix} 0.9 \\ 0.9 \\ 1.9 \end{bmatrix}$$

Example

- we iteratively apply the following equation ($i = 1$)

$$Q(s,a) \coloneqq R(s,a) + \gamma \sum_{s' \epsilon S} P(s'|s,a)V(s'),\ \forall s, \forall a$$

$$Q(1, left) \coloneqq 0 + 0.9(1\times0.9 + 0\times0.9 + 0\times1.9) = 0.81$$
$$Q(2, left) \coloneqq 0 + 0.9(0.8\times0.9 + 0.2\times0.9 + 0\times1.9) = 0.81$$
$$Q(3, left) \coloneqq 1 + 0.9(0\times0.9 + 0.8\times0.9 + 0.2\times1.9) = 1.99$$

$$Q(1, right) \coloneqq 0 + 0.9(0.2\times0.9 + 0.8\times0.9 + 0\times1.9) = 0.81$$
$$Q(2, right) \coloneqq 0 + 0.9(0\times0.9 + 0.2\times0.9 + 0.8\times1.9) = 1.53$$
$$Q(3, right) \coloneqq 1 + 0.9(0\times0.9 + 0\times0.9 + 1\times1.9) = 2.71$$

Example

- we iteratively apply the following equations ($i = 1$)

$$Q(s,a) \coloneqq R(s,a) + \gamma \sum_{s' \epsilon S} P(s'|s,a)V(s'),\ \ \forall s, \forall a$$

$$Q = \begin{bmatrix} 0.81 & 0.81 \\ 0.81 & 1.53 \\ 1.99 & 2.71 \end{bmatrix}$$

$$V(s) \coloneqq \max_{a \in A} Q(s,a), \forall s$$

$$V = \begin{bmatrix} 0.81 \\ 1.53 \\ 2.71 \end{bmatrix}$$

Example

- we iteratively apply the following equations ($i = 2$)

$$Q(s,a) \coloneqq R(s,a) + \gamma \sum_{s' \epsilon S} P(s'|s,a)V(s'),\ \ \forall s, \forall a$$

$$Q = \begin{bmatrix} 0.73 & 1.25 \\ 0.86 & 2.23 \\ 2.59 & 3.44 \end{bmatrix}$$

$$V(s) \coloneqq \max_{a \in A} Q(s,a), \forall s$$

$$V = \begin{bmatrix} 1.25 \\ 2.23 \\ 3.44 \end{bmatrix}$$

Example

- After a few iterations…

Example

- we iteratively apply the following equations ($i = 47$)

$$Q(s,a) := R(s,a) + \gamma \sum_{s' \epsilon S} P(s'|s,a)V(s'),\ \forall s, \forall a$$

$$Q = \begin{bmatrix} 6.88 & 7.65 \\ 7.07 & 8.72 \\ 9.06 & 9.94 \end{bmatrix}$$

$$V(s) := \max_{a \in A} Q(s,a), \forall s$$

$$V = \begin{bmatrix} 7.65 \\ 8.72 \\ 9.94 \end{bmatrix}$$

Example

- we iteratively apply the following equations ($i = 48$)

$$Q(s,a) \coloneqq R(s,a) + \gamma \sum_{s' \epsilon S} P(s'|s,a) V(s'),\ \forall s, \forall a$$

$$Q = \begin{bmatrix} 6.89 & 7.66 \\ 7.08 & 8.73 \\ 9.07 & 9.95 \end{bmatrix}$$

$$V(s) \coloneqq \max_{a \in A} Q(s,a), \forall s$$

$$V = \begin{bmatrix} 7.66 \\ 8.73 \\ 9.95 \end{bmatrix}$$

Example

- The algorithm repeated until V did not change significantly between two consecutive steps.

- Result:

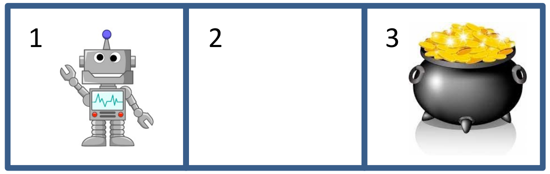
$$V^* = \begin{bmatrix} 7.66 \\ 8.73 \\ 9.95 \end{bmatrix}$$

$$Q^* = \begin{bmatrix} 6.89 & 7.66 \\ 7.08 & 8.73 \\ 9.07 & 9.95 \end{bmatrix}$$

Example

- After computing the optimal Q^*, we can extract the policy as follows:

$$\pi^*(s) \in \underset{a \in A}{\operatorname{argmax}} Q^*(s, a)$$



$$\pi^* = \begin{bmatrix} right \\ right \\ right \end{bmatrix}$$

$$\pi^* = \begin{bmatrix} 0 & 1 \\ 0 & 1 \\ 0 & 1 \end{bmatrix}$$

Python Code

```python
import numpy as np
np.set_printoptions(precision=2, suppress=True)

# States
S = ['1', '2', '3']

# Actions
A = ['L', 'R']

# Transition probabilities

L = np.array([[1.0, 0.0, 0.0],
              [0.8, 0.2, 0.0],
              [0.0, 0.8, 0.2]])

R = np.array([[0.2, 0.8, 0.0],
              [0.0, 0.2, 0.8],
              [0.0, 0.0, 1.0]])

P = [L, R]

# Reward function

R = np.array([[0.0, 0.0],
              [0.0, 0.0],
              [1.0, 1.0]])

gamma = 0.9
```

Python Code

```
# Initialize V
V = np.ones(len(S))

err = 1
i = 0

while err > 1e-2:

    Q = []

    # Compute Q-values associated with each action
    for a in range(len(A)):
        Q += [R[:, a] + gamma * P[a].dot(V)]

    print('Q values at time ',i)
    print(Q)

    # Compute maximum for each state
    Vnew = np.max(Q, axis=0)
    print('V-function at time ',i)
    print(Vnew)

    # Compute error
    err = np.linalg.norm(V - Vnew)

    # Update V
    V = Vnew

    i += 1


print('\nV* =')
print(V[:, None])

print('\nQ* =')
print(Q)
```

Thank You

sardinha@inf.puc-rio.br