MDP: Markov Decision Problem

Deep Reinforcement Learning

Alberto Sardinha
sardinha@inf.puc-rio.br

Outline

- **Markov decision problem**
- Policy iteration
- Example

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

Outline

- Markov decision problem
- **Policy iteration**
- Example

Markov Decision Problem (MDP)

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

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

Markov Decision Problem (MDP)

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

If we write the expression above using vector notation:

$$V^{\pi} = R_{\pi} + \gamma P_{\pi} V^{\pi}$$

Markov Decision Problem (MDP)

$$V^{\pi} = R_{\pi} + \gamma P_{\pi} V^{\pi}$$

$$V^{\pi} - \gamma P_{\pi} V^{\pi} = R_{\pi}$$

$$(I - \gamma P_{\pi}) V^{\pi} = R_{\pi}$$

$$\boxed{V^{\pi} = (I - \gamma P_{\pi})^{-1} R_{\pi}}$$

Policy Evaluation of π

Markov Decision Problem (MDP)

$$\pi_g(s) = \operatorname*{argmax}_{a \in A} Q(s,a)$$

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

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

Policy Improvement

Markov Decision Problem (MDP)

Why Policy Improvement?

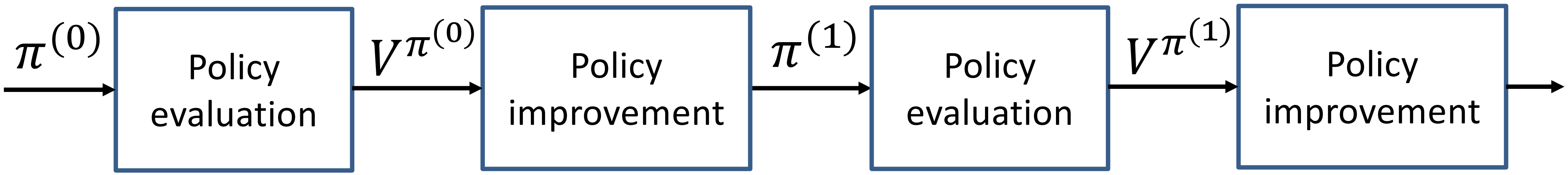
- Given a policy π
- Given a state-value function V^{π}
- We have:

$$\boxed{V^{\pi_g} \geq V^{\pi}}$$

Policy π_g is better than π

Policy Iteration

This leads us to a new algorithm called Policy Iteration:

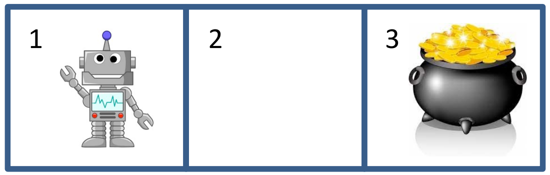


Outline

- Markov decision problem
- Policy iteration
- **Example**

Example

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

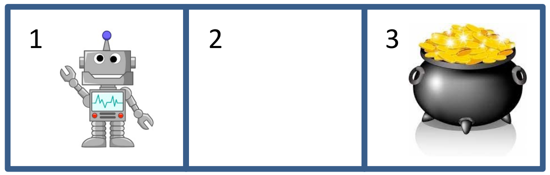


Example

- 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 policy iteration:
  - We start with the following policy:

$$\pi(a|s) = \begin{bmatrix} 0.5 & 0.5 \\ 0.5 & 0.5 \\ 0.5 & 0.5 \end{bmatrix}$$

Example

- We evaluate the policy ($i = 0$):

$$P_\pi(s'|s) = \sum_{a \in A} \pi(a|s) P(s'|s, a)$$

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

$$V^\pi = (I - \gamma P_\pi)^{-1} R_\pi$$

Example

- We evaluate the policy ($i = 0$):

$$P_\pi = \begin{bmatrix} 0.6 & 0.4 & 0 \\ 0.4 & 0.2 & 0.4 \\ 0 & 0.4 & 0.6 \end{bmatrix}$$

$$R_\pi = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$$

$$V^\pi = \begin{bmatrix} 2.39 \\ 3.05 \\ 4.56 \end{bmatrix}$$

Example

- We improve the policy ($i = 0$):

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

$$\pi_g(s) = \underset{a \in A}{\operatorname{argmax}} \, Q(s,a)$$

Convert $\pi_g(s)$ to $\pi_g(s|a)$

$$(\pi_g == \pi)?$$

Example

- We improve the policy ($i = 0$):

$$Q(s,a) = \begin{bmatrix} 2.15 & 2.63 \\ 2.27 & 3.83 \\ 4.02 & 5.11 \end{bmatrix}$$

$$\pi_g(s|a) = \begin{bmatrix} 0 & 1 \\ 0 & 1 \\ 0 & 1 \end{bmatrix}$$

Example

- We evaluate the policy ($i = 1$):

$$P_\pi(s'|s) = \sum_{a \in A} \pi(a|s) P(s'|s,a)$$

$$R_\pi(s) = \sum_{a \in A} \pi(a|s) R(s,a)$$

$$V^\pi = (I - \gamma P_\pi)^{-1} R_\pi$$

Example

- We evaluate the policy ($i = 1$):

$$P_\pi = \begin{bmatrix} 0.2 & 0.8 & 0 \\ 0 & 0.2 & 0.8 \\ 0 & 0 & 1 \end{bmatrix}$$

$$R_\pi = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$$

$$V^\pi = \begin{bmatrix} 7.71 \\ 8.78 \\ 10 \end{bmatrix}$$

Example

- We improve the policy ($i = 1$):

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

$$\pi_g(s) = \operatorname*{argmax}_{a \in A} Q(s,a)$$

Convert $\pi_g(s)$ to $\pi_g(s|a)$

$$(\pi_g == \pi)?$$

Example

- We improve the policy ($i = 1$):

$$Q(s,a) = \begin{bmatrix} 6.94 & 7.71 \\ 7.13 & 8.78 \\ 9.12 & 10 \end{bmatrix}$$

$$\pi(s|a) = \begin{bmatrix} 0 & 1 \\ 0 & 1 \\ 0 & 1 \end{bmatrix}$$

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

```
def evaluate(Rw, P, pol):
    """ evaluate(Rw, P, pol) computes the state-value function associated with policy pol.

        :param Rw: 2D np.ndarray with |S| rows and |A| columns; element Rw[s,a] is the reward of action a in state s
        :param P: 3D np.ndarray with |A| |S| x |S| matrices; matrix P[a] is the transition matrix associated with
         action a.
        :param pol: 2D np.ndarray with the same dimension as Rw; element pol[s,a] is the probability of action a in
         state s.

        :return V: 2D np.ndarray with |S| elements, where element s is Vpi(s)."""

    # Problem dimensions
    nS, nA = Rw.shape

    # Policy-averaged reward
    Rpi = (pol * Rw).sum(axis = 1)

    print('Rpi =')
    print(Rpi)

    # Policy-averaged probabilities
    Ppi = pol[:, 0, None] * P[0]

    for a in range(1, nA):
        Ppi += pol[:, a, None] * P[a]

    print('Ppi =')
    print(Ppi)

    # Use matrix inversion to compute Vpi
    Vpi = np.linalg.inv(np.eye(nS) - gamma * Ppi).dot(Rpi)

    return Vpi[:, None]

# -- End: evaluate
```

Python Code

```python
import time

# Initialize policy

pol = np.ones((len(S), len(A))) / len(A)

print('Initial policy =')
print(pol)

# Auxiliary matrix to store temporary Q-values
Q = np.zeros((len(S), len(A)))

quit = False
i = 0

t = time.time()

while not quit:
    # Evaluate policy
    V = evaluate(Rw, P, pol)

    print('V =')
    print(V)

    # Compute Q-values
    for a in range(len(A)):
        Q[:, a, None] = Rw[:, a, None] + gamma * P[a].dot(V)

    print('Q =')
    print(Q)

    # Compute maximizing policy
    Qmax = Q.max(axis=1, keepdims=True)
    polnew = np.isclose(Q, Qmax, atol=1e-10, rtol=1e-10).astype(int)
    polnew = polnew / polnew.sum(axis = 1, keepdims = True)

    print('Policy =')
    print(polnew)
```

Python Code

```
    quit = (pol == polnew).all()
    pol = polnew

    i += 1

t = time.time() - t

print('N. iterations:', i)
print('Time taken:', round(t, 3), 'seconds')
print('\npi*=')
print(pol)
print('\nV* =')
print(V[:, None])
```

Thank You

sardinha@inf.puc-rio.br