Data-driven and Learning-based Control

4th hands-on session

Erica Salvato

Table of Contents

Reinforcement Learning taxonomy

We can classify Reinforcement Learning approaches in:

- **Value-function methods:**
  The policy is implicitly defined via $V(x^{(k)})$ or $Q(x^{(k)}, u^{(k)}) \rightarrow$ **Critic**

- **Policy optimization methods:**
  The policy is a parameterized function whose weights are learned in order to maximize the expected cumulative discounted reward$\rightarrow$ **Actor**

- **Actor-critic methods:**
  Merging the two ideas by guiding the actor's learning on the basis of the critic's estimated return

Value-function methods taxonomy

We can classify Value-function methods:

- depending on the **state set representation**:
  - Tabular
  - Function approximation
- depending on **how the policy is used in the evaluation/improvement steps**:
  - On-policy
  - Off-policy

Value-function methods taxonomy

We can classify Value-function methods:

- depending on the **state set representation**:
  - Tabular
  - Function approximation
- depending on **how the policy is used in the evaluation/improvement steps**:
  - On-policy
  - Off-policy

Value function methods assume to work with a discrete compact action set, namely

$$\mathcal{U} = \{u_1, u_2, \ldots, u_h\}$$

On-policy vs. Off-policy

- On-policy means that the evaluation is based on the **current policy**:

$$Q\left(x^{(k)}, u^{(k)}\right) = Q\left(x^{(k)}, u^{(k)}\right) + \alpha\left[r^{(k+1)} + \gamma Q\left(x^{(k+1)}, u^{(k+1)}\right) - Q\left(x^{(k)}, u^{(k)}\right)\right]$$

- Off-policy means that the evaluation is based on the **greedy policy**:

$$Q\left(x^{(k)}, u^{(k)}\right) = Q\left(x^{(k)}, u^{(k)}\right) + \alpha\left[r^{(k+1)} + \gamma \max Q\left(x^{(k+1)}, u\right) - Q\left(x^{(k)}, u^{(k)}\right)\right]$$

SARSA algorithm

Initialization. $Q \,\forall x \in \mathcal{X}\; u \in \mathcal{U}$

Repeat (for each episode)

- Set $x^{(0)}$
- Select an action $u^{(k)}$ with ϵ-greedy policy
- Repeat for each step of the episode until the terminal condition is met
  - Perform the action $u^{(k)}$, observe $x^{(k+1)}$ and $r^{(k+1)}$
  - Select an action $u^{(k+1)}$ with ϵ-greedy policy
  - $Q\left(x^{(k)}, u^{(k)}\right) = Q\left(x^{(k)}, u^{(k)}\right) + \alpha\left[r^{(k+1)} + \gamma Q\left(x^{(k+1)}, u^{(k+1)}\right) - Q\left(x^{(k)}, u^{(k)}\right)\right]$
  - Update $x^{(k)} = x^{(k+1)}$, $u^{(k)} = u^{(k+1)}$

Q-Learning algorithm

Initialization. $Q \, \forall x \in \mathcal{X} \; u \in \mathcal{U}$

Repeat (for each episode)

- Set $x^{(0)}$
- Repeat for each step of the episode until the terminal condition is met
  - Select an action $u^{(k)}$ with ϵ-greedy policy
  - Perform the action $u^{(k)}$, observe $x^{(k+1)}$ and $r^{(k+1)}$
  - $Q\left(x^{(k)}, u^{(k)}\right) = Q\left(x^{(k)}, u^{(k)}\right) + \alpha \left[r^{(k+1)} + \gamma \max Q\left(x^{(k+1)}, u\right) - Q\left(x^{(k)}, u^{(k)}\right)\right]$
  - Update $x^{(k)} = x^{(k+1)}$,

Tabular vs. Function approximation

- Tabular means that the $Q\left(x^{(k)}, u^{(k)}\right)$ representation is a lookup table, i.e., one entry for every state/action pair.

  Consequently, it assumes to work with a discrete compact state set.

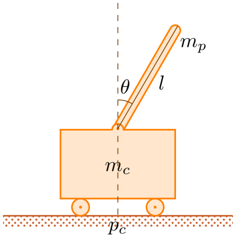
	u_1	u_2	$\dots$	u_h
x_1	$Q(x_1, u_1)$	$Q(x_1, u_2)$	$\dots$	$Q(x_1, u_h)$
x_2	$Q(x_2, u_1)$	$Q(x_2, u_2)$	$\dots$	$Q(x_2, u_h)$
$\dots$	$\dots$	$\dots$	$\dots$	$\dots$
x_l	$Q(x_l, u_1)$	$Q(x_l, u_2)$	$\dots$	$Q(x_l, u_h)$

- Function approximation means to define an approximate representation of the value function $\hat{V}(x)$ or of the action-value function $\hat{Q}(x, u)$.

$$Q(x, u) \approx \hat{Q}_\theta(x, u)$$

Table of Contents

The pole balancing problem



$$\ddot{\theta} = \frac{g\sin(\theta) + \cos(\theta)\left[\frac{-F - m_p\, l\,\dot{\theta}^2\sin(\theta)}{m_c + m_p}\right] - \frac{\mu_p \dot{\theta}}{m_p\, l}}{l\left[\frac{4}{3} - \frac{m_p\, cos^2(\theta)}{m_c + m_p}\right]}$$

$$\ddot{p}_c = \frac{F + m_p\, l\left[\dot{\theta}^2\,\sin(\theta) - \ddot{\theta}\,\cos(\theta)\right]}{m_c + m_p}$$

Hands-on outline

Given the cart and pole system previously defined, we would like to apply tabular Q-Learning with the goal of balancing the cart and pole.

1. Given the $\mathcal{X}, \mathcal{U}$ of the first hands-on define
   - the set of possible initial conditions of an episode
   - the terminal conditions of the episode

Expectation: Argue the answer

Hands-on outline

2 4th hands-on session

Given the cart and pole system previously defined, we would like to apply tabular Q-Learning with the goal of balancing the cart and pole.

1. Given the $\mathcal{X}, \mathcal{U}$ of the first hands-on define
   - the set of possible initial conditions of an episode
   - the terminal conditions of the episode

Expectation: Argue the answer

2. Provide a proper discretization of the state space and define the size of the Q

Expectation: Argue the answer

Hands-on outline

2 4th hands-on session

3. Create a code able to select the discretized version of the state given a measured state of the system

Expectation: Create a code able to convert measured states of the cart and pole system into the discretized version (Notice that we need it only to access the correct position in the table).

Hands-on outline

2 4th hands-on session

3. Create a code able to select the discretized version of the state given a measured state of the system

Expectation: Create a code able to convert measured states of the cart and pole system into the discretized version (Notice that we need it only to access the correct position in the table).

4. Given the terminal conditions of the episode create a code able to perform a fixed number of episodes of a fixed number of steps starting from random initial conditions and applying random inputs

Expectation: Create a code able to simulate the cart and pole for different episodes of a fixed number of steps, where each episode starts with a different initial condition randomly chosen. (Notice that terminal condition means that the episode terminates even if the number of time-steps have not been performed)

Hands-on outline

5. Create a function able to perform ϵ-greedy policy for a chosen ϵ

Expectation: Given the ϵ and the Q the code should return the action to be performed following ϵ-greedy policy

Hands-on outline

5. Create a function able to perform ϵ-greedy policy for a chosen ϵ

Expectation: Given the ϵ and the Q the code should return the action to be performed following ϵ-greedy policy

6. Create a code that applies Tabular Q-Learning algorithm

Expectation: Properly select the parameters α, γ, ϵ and motivate the choice. The code should use all the results of the previous steps in order to construct the Tabular Q-Learning algorithm for balancing the cart and pole system. Plot the cumulative discounted reward

Hands-on outline

2 4th hands-on session

7. Copy and past your Q-learning algorithm and apply the changes needed to convert it into a SARSA algorithm

Expectation: The code should use all the results of the previous steps in order to construct the Tabular SARSA algorithm for balancing the cart and pole system. Plot the cumulative discounted reward

Questions’ time!