Data-driven and Learning-based Control

5th 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

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

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	...	u_h
x_1	$Q(x_1, u_1)$	$Q(x_1, u_2)$	...	$Q(x_1, u_h)$
x_2	$Q(x_2, u_1)$	$Q(x_2, u_2)$	...	$Q(x_2, u_h)$
...	...	...	...	...
x_l	$Q(x_l, u_1)$	$Q(x_l, u_2)$	...	$Q(x_l, u_h)$

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

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

Linear function approximation

In particular, approximated Q-values are computed as follows:

$$\hat{Q}_\theta(x,u) = \sum_{l=1}^{n} \theta_l^\top \phi_l(x,u) = \theta^\top \phi(x,u)$$

where $\phi(x,u) = \begin{bmatrix} \phi_1(x,u) \\ \phi_2(x,u) \\ \vdots \\ \phi_h(x,u) \end{bmatrix}$ and $\theta = \begin{bmatrix} \theta_1 \\ \theta_2 \\ \vdots \\ \theta_h \end{bmatrix}$

Linear function approximation

Since in Value-function methods we deal with discrete compact action set, assuming that it includes c values $\{u_1, \ldots, u_c\}$, for approximating Q-values we have the following:

$$\phi(x, u_i) = [0, \ldots, 0, \ldots, \phi_1(x), \ldots, \phi_n(x), 0, \ldots, 0]$$

$$u_1 \quad \ldots \quad u_i \quad u_c$$

and

$$\theta = \begin{bmatrix} \theta_1 & \theta_2 & \ldots & \theta_{h*c} \end{bmatrix}$$

Radial Basis Functions

1 A brief recap

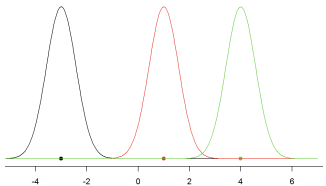
A Radial basis function is a bell-shaped (Gaussian) function used as basis function.

The general form of a Radial basis function given an n-dimensional state vector is given by:

$$\phi_i(x) = \exp\left(-\frac{||x - c_i||^2}{2\sigma_i^2}\right)$$

where:

- c_i is a vector of n components whose values are the centers of the i-th BF.
- σ_i is a positive real value corresponding to the width of the i-th BF.



SARSA with linear function approximation

Initialization. θ choose $\phi(x, u)$

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
  - $Q\left(x^{(k)}, u^{(k)}\right) = \theta^\top \phi\left(x^{(k)}, u^{(k)}\right)$ and $Q\left(x^{(k+1)}, u^{(k+1)}\right) = \theta^\top \phi\left(x^{(k+1)}, u^{(k+1)}\right)$
  - $\theta = \theta + \alpha\left[r^{(k+1)} + \gamma Q\left(x^{(k+1)}, u^{(k+1)}\right) - Q\left(x^{(k)}, u^{(k)}\right)\right] \nabla_\theta Q\left(x^{(k)}, u^{(k)}\right)$
  - Update $x^{(k)} = x^{(k+1)}$, $u^{(k)} = u^{(k+1)}$

Q-Learning with linear function approximation

Initialization. θ choose $\phi(x, u)$

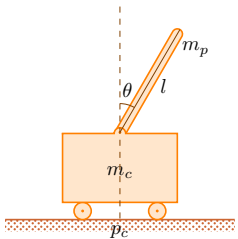
Repeat (for each episode)

- 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 $r^{(k+1)}$
  - $Q\left(x^{(k)}, u^{(k)}\right) = \theta^\top \phi\left(x^{(k)}, u^{(k)}\right)$ and $\forall u\, Q\left(x^{(k+1)}, u\right) = \theta^\top \phi\left(x^{(k+1)}, u\right)$
  - $\theta = \theta + \alpha\left[r^{(k+1)} + \gamma \max Q\left(x^{(k+1)}, u\right) - Q\left(x^{(k)}, u^{(k)}\right)\right] \nabla_\theta Q\left(x^{(k)}, u^{(k)}\right)$
  - Update $x^{(k)} = x^{(k+1)}$,

Table of Contents

The pole balancing problem

2 4th hands-on session



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

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

Hands-on outline

2 4th hands-on session

Given the cart and pole system previously defined, we would like to apply Q-Learning with RBF linear function approximation to balance the cart and pole.

1. Given the $\mathcal{X}$ set, select centers and standard deviation of RBFs and create a code that given a state x returns $\phi(x, \cdot)$.

Expectation: Set properly centers of each RBF and variance. Create a code able to return $\phi(x, \cdot)$ for all u given a specific x. (N.B. In choosing centers, try to take a guideline from the discretization of previous hands-on)

Hands-on outline

2 4th hands-on session

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

Expectation: Properly select the parameters α, γ, ϵ and motivate the choice. The code should be an adaptation of the code of the previous hands-on to construct the Q-Learning algorithm with RBFs linear function approximation to balance the cart and pole system. Exploit the code of the previous point. Plot the cumulative discounted reward

Hands-on outline

2 4th hands-on session

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

Expectation: Properly select the parameters α, γ, ϵ and motivate the choice. The code should be an adaptation of the code of the previous hands-on to construct the Q-Learning algorithm with RBFs linear function approximation to balance the cart and pole system. Exploit the code of the previous point. Plot the cumulative discounted reward

3. Copy and paste 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 SARSA algorithm with RBFs linear function approximation to balance the cart and pole system. Plot the cumulative discounted reward

Questions' time!

UNIVERSITÀ
DEGLI STUDI
DI TRIESTE