Q-learning on Linear Quadratic Problem

LINKÖPINGS UNIVERSITET

Farnaz Adib Yaghmaie

Linkoping University, *Sweden*
farnaz.adib.yaghmaie@liu.se

April 6, 2021

- **Dynamics:**

$$s_{t+1} = As_t + Bu_t + w_t$$

- **State and action:**

$$s_t \in \mathbb{R}^n, \\ u_t \in \mathbb{R}^m$$

- **Cost function ($\equiv$ negative of reward):**

$$c_t = s_t^\dagger Q s_t + u_t^\dagger R u_t, \quad Q \geq 0, R > 0$$

Solvability Criterion: Minimize $V(s)$ with respect to the policy π

$$V(s_t) = \mathbf{E}[\sum_{k=t}^{+\infty}(c_k - \lambda)|s_t]$$

where λ is the average cost

$$\lambda = \lim_{T\to\infty} \frac{1}{T}\sum_{t=1}^{T} c_t$$

Note that minimizing $V(s_t)$ and λ are equivalent.

The agents learn a quadratic Q function

$$Q(s,a) = \begin{bmatrix} s^\dagger & a^\dagger \end{bmatrix} \begin{bmatrix} g_{ss} & g_{sa} \\ g_{sa}^\dagger & g_{aa} \end{bmatrix} \begin{bmatrix} s \\ a \end{bmatrix} = z^\dagger G z \tag{1}$$

The policy is given by optimizing the Q function

$$\pi = -g_{aa}^{-1} g_{sa}^\dagger \, s = Ks \tag{2}$$

1. Compute the empirical average cost $\lambda = \frac{1}{T}\sum_{t=1}^{T} c_t$
2. Collect data
   - Observe s and select a

$$a = Ks + r, \quad r \sim \mathcal{N}(0, \sigma^2).$$

   - Apply a and observe c, s'.
   - Add s, a, c, s' to the history.
3. Estimated the kernel of Q by Least Squares Temporal Difference (LSTD)

$$vecs(G) = (\tfrac{1}{T}\sum_{t=1}^{T}\Psi_t(\Psi_t - \Psi_{t+1})^{\dagger})^{-1}(\tfrac{1}{T}\sum_{t=1}^{T}\Psi_t(c_t - \lambda)) \tag{3}$$

where

$$z = \begin{bmatrix} s \\ a \end{bmatrix}, \ \Psi = [z_1^2, 2z_1z_2, \ldots, 2z_1z_n, z_2^2, \ldots,\ 2z_2z_n,\ \ldots, z_n^2]^{\dagger}.$$

- **Dynamics:**

$$s_{t+1} = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix} s_t + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u_t + w_t$$

- **Cost function ($\equiv$ negative of reward):**

$$c_t = s_t^\dagger \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} s_t + u_t^\dagger 1 u_t.$$

- **Exact analytical solution assuming full information about dynamics**

$$u_t^* = \begin{bmatrix} -0.422 & -1.244 \end{bmatrix} s_t$$

- **Initialization of the algorithm**

$$u_t = \begin{bmatrix} -0.616 & -1.614 \end{bmatrix} s_t$$

Try the following:

- Run

  Crash_course_on_RL/q_on_lq_notebook.ipynb

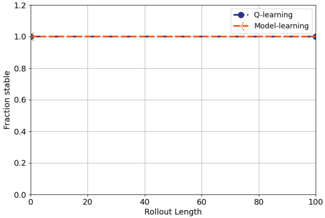
  and verify the median of the error in K is $\sim 0.01\%$.

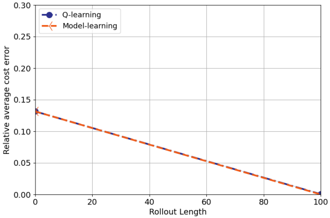
- Set

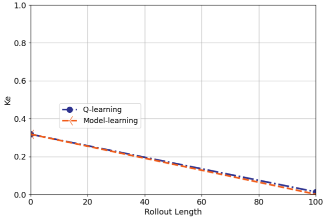
  'explore_mag=0.0' in 'My_q_learning.ql'

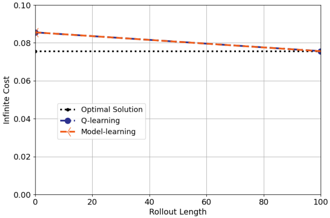
  and verify that the agent cannot solve the problem! you don't get any stable controller.

- Make sure you understand the code!


Q-learning
Model-learning
Fraction stable
1.2
1.0
0.8
0.6
0.4
0.2
0.0
0
20
40
60
80
100
Rollout Length


Q-learning
Model-learning
Relative average cost error
0.30
0.25
0.20
0.15
0.10
0.05
0.00
0
20
40
60
80
100
Rollout Length


Q-learning
Model-learning
Ke
1.0
0.8
0.6
0.4
0.2
0.0
0
20
40
60
80
100
Rollout Length


Optimal Solution
Q-learning
Model-learning
Infinite Cost
0.10
0.08
0.06
0.04
0.02
0.00
0
20
40
60
80
100
Rollout Length

Important observations

- Q-learning performs superb on the LQ problem
- No hyper-parameters to tune

Email your questions to

farnaz.adib.yaghmaie@liu.se