Policy Gradient on Linear Quadratic Problem

Farnaz Adib Yaghmaie

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

March 12, 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 the average cost ($\equiv$ maximize the average reward)

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

We consider a linear policy, so the mean of the pdf is selected as

$$\mu_\theta(s) = \theta\, s \tag{1}$$

and the pdf is given by

$$\pi_\theta(s) = \frac{1}{\sqrt{(2\pi\sigma^2)^{n_a}}} \exp[-\frac{1}{2\sigma^2}(a-\theta s)^\dagger (a-\theta s)]$$

1. Collect data
   - Observe s and sample $a \sim \pi_\theta(s)$

```
a = theta s + sigma * np.random.randn(n_a)
```

   - Apply a and observe r.
   - Add s, a, r to the history.
2. Update the parameter θ
   - We calculate the reward and standardize it.
   - We calculate the gradient using

$$\nabla_\theta J = \frac{1}{\sigma^2 |\mathcal{D}|} \sum_{\tau \in \mathcal{D}} \sum_{t=1}^{T} (a_t - \theta\, s_t) s_t^\dagger R(T). \tag{2}$$

   - We optimize the policy by a gradient algorithm (e.g. an ADAM optimizer)

- **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/pg_on_lq_notebook.ipynb

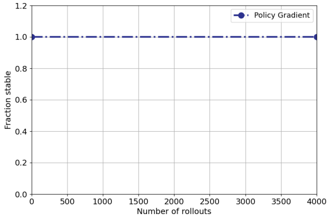
  and verify the median of the error in estimating the optimal gain is $\sim 0.08\%$.

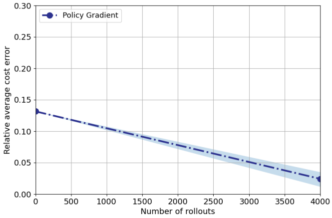
- Set

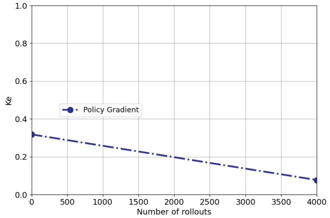
  'explore_mag=0.000001' in 'Mypgrl.pg_linpolicy'

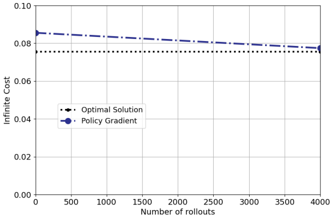
  and verify that the agent cannot learn the optimal gain by using a deterministic policy in PG.

- Make sure you understand the code!


Policy Gradient
Fraction stable
1.2
1.0
0.8
0.6
0.4
0.2
0.0
0
500
1000
1500
2000
2500
3000
3500
4000
Number of rollouts


Policy Gradient
Relative average cost error
0.30
0.25
0.20
0.15
0.10
0.05
0.00
0
500
1000
1500
2000
2500
3000
3500
4000
Number of rollouts


Policy Gradient
Ke
1.0
0.8
0.6
0.4
0.2
0.0
0
500
1000
1500
2000
2500
3000
3500
4000
Number of rollouts


Optimal Solution
Policy Gradient
Infinite Cost
0.10
0.08
0.06
0.04
0.02
0.00
0
500
1000
1500
2000
2500
3000
3500
4000
Number of rollouts

Email your questions to

farnaz.adib.yaghmaie@liu.se