Reinforcement Learning Lecture: Homework 09

Ngo Anh Vien

MLR, University of Stuttgart

June 13, 2016

1 Exercise 01

[Programming] Write a policy gradient algorithm (using Finite Difference Method) on Cart-Pole as in Fig. 1 (Sutton's RL book, 1998). The task is to apply forces to a cart moving a long a track in order to keep the pole balanced. If the pole falls apart a given angle (12 degree = 0.21 rad), the episode terminates. The termination also happens when the cart runs off the track. The state space of this task is defined as $s = \{p, \dot{p}, \phi, \dot{\phi}\}$, where $p, \dot{p}$ are the position and velocity of the cart, $p \in [-2.4, 2.4]$; $\phi, \dot{\phi}$ are the the angle and angular velocity (w.r.t the vetical) of the pole. The episode always starts at $\{0, 0, 0, 0\}$. Actions are continuous $a \in [-10, +10]$. The reward function is simple, if the pole is still balanced $r = 1.$, otherwise if fails $r = -1.$. For the dynamics of this system, you can refer to the code website of Sutton's book (see this link), in which the dynamics is deterministic. To make the task more interesting, I added a small Gaussian noise to the update the p and ϕ as you see in the code provided.

Implmentation Note:

The policy is stochastic Gaussian controller as

$$\pi(a|s) = \frac{1}{Z} \exp\Big(-\frac{(\theta^\top s - a)^2}{2\sigma^2}\Big)$$

where Z is a normalization constant of the Gaussian policy distribution, and note that we are using a linear policy function purturbed with a small Gaussian noise (which is similar to a PID controller).

- The code of environment is given. Put the folder Ex09 into "examples" folder (as similar to the code base given in Ex02)

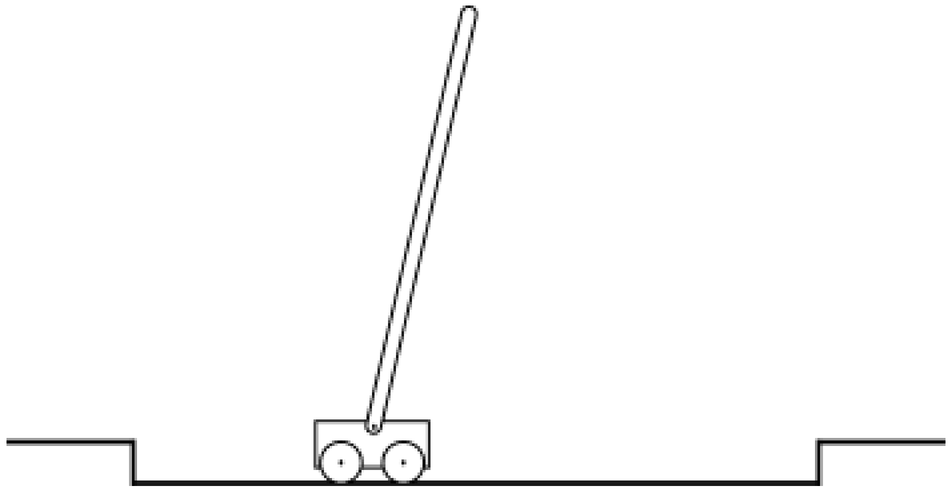

Figure 1: A pole-balancing task

- See the FD algorithm in slide 22 (an updated version).
- Let's fix $\sigma^2 = 0.5$.
- Each $\delta\theta_i^{(j)}$ (the dimension j of a sample i) is sampled from a uniform distribution $[0, 0.2]$
- $\gamma = 1.0$
- The number of samples of $\delta\theta$: $M = 20$.
- The number of samples to evaluate each $J(\theta) = 20$
- The step-size $\alpha = 0.5$
- Use normalized gradients

$$\theta = \theta + \alpha \frac{g_{FD}}{\|g_{FD}\|}$$

- Each episode terminates either after 5000 steps or observing the failure of the pole.

1.1 Results for report

Step by step:

- Run the above algorithm for 200 iterations. Record $\{k, J(\theta_k)\}_{k=1}^{100}$ for each iteration.
- Plot the data $\{k, J(\theta_k)\}_{k=1}^{100}$

1.2 Adaptive step-size

In more complex domain, we might need to adaptively tune α. In this question, you are asked to program the Rprop method (Resilient Back Propagation)(see the algorithm below) to tune α at each iteration. Assuming that $\theta \in R^n$, the following algorithm tune the step-size for each dimension,

---

for $i = 1 : n$ **do**
  if $g_{FD}^{(i)} g_{prev}^{(i)} > 0$ **then**
    $\alpha_i = 1.2\alpha_i$
    $\theta_i = \theta_i + \alpha_i \ \text{sign}(g_{FD}^{(i)})$
    $g_{prev}^{(i)} = g_{FD}^{(i)}$
  else if $g_{FD}^{(i)} g_{prev}^{(i)} < 0$ **then**
    $\alpha_i = 0.5\alpha_i$
    $\theta_i = \theta_i + \alpha_i \ \text{sign}(g_{FD}^{(i)})$
    $g_{prev}^{(i)} = 0$
  else
    $\theta_i = \theta_i + \alpha_i \ \text{sign}(g_{FD}^{(i)})$
    $g_{prev}^{(i)} = g_{FD}^{(i)}$
  end if
  optionally: cap $\alpha_i \in [\alpha_{\min} \ \theta_i, \alpha_{\max} \ \theta_i]$
end for

Implementation note:

- Initialize $\alpha_i = 0.5$, $g_{prev}^{(i)} = 0$

2 Exercise 02

How do you design a policy gradient algorithm for a discrete domain?