ECE-GY 9123 / Spring 2021

Homework 5

Please upload your assignments on or before April 26, 2021.

- You are encouraged to discuss ideas with each other; but
- you **must acknowledge** your collaborator, and
- you **must compose your own** writeup and/or code independently.
- We **require** answers to theory questions to be written in LaTeX, and answers to coding questions in Python (Jupyter notebooks)
- Upload your answers in the form of a single PDF on Gradescope.

---

1. **(3 points)** *Policy gradients*. In class we derived a general form of policy gradients. Let us consider a special case here. Suppose the step size is η. We consider where past actions and states do not matter; different actions a_i give rise to different rewards R_i.

   a. Define the mapping π such that $\pi(a_i) = \text{softmax}(\theta_i)$ for $i = 1, \ldots, k$, where k is the total number of actions and θ_i is a scalar parameter encoding the value of each action. Show that if action a_i is sampled, then the change in the parameters in REINFORCE is given by:
   $$\Delta\theta_i = \eta R_i(1 - \pi(a_i)).$$

   b. Intuitively explain the dynamics of the above gradient updates.

   Solution

   a. According to REINFORCE, the update is
   $$\theta \leftarrow \theta - \eta R(\tau) \frac{\partial}{\partial \theta} \log \pi(\tau).$$
   In this case, the trajectory does not matter, and the reward is just the function of the current action, i.e., $R(\tau) = R(a_i) = R_i$. Moreover, by definition we have:
   $$\begin{aligned} \partial \log \pi(a_i) &= \partial \left(\theta_i - \log(\sum \exp \theta_i) \right) \\ &= 1 - \text{softmax}(\theta_i). \end{aligned}$$
   Therefore, dropping the step size, we get the update direction as:
   $$R_i(1 - \pi(a_i)).$$

   b. This is unlike gradient descent where the change in the parameters is (roughly) proportional to the prediction error, and the weights stabilize when the error roughly goes to zero. Here we observe the change in parameters is negatively correlated with the probability of sampling actions. Intuitively, a) large steps occur when the probability of a given action is low, but its reward is high; and b) if the reward of a given action is low then the weights don't change that much. Eventually, the algorithm approximately stabilizes when high-reward actions also get assigned high probabilities.

2. **(3 points)** *Designing rewards in Q-learning*. Suppose we are trying to solve a maze with a goal and a (stationary) monster in some location, and the goal is to reach the goal in the minimum number of moves. We are tasked with designing a suitable reward function for Q-learning. There are two options:

   a. We declare a reward of +2 for reaching the goal, -1 for running into a monster, and 0 for every other move.

   b. We declare a reward of +1.5 for reaching the goal, -1.5 for running into a monster, and -0.5 for every other move.

   Which of these reward functions might lead to better policies?

   (*Hint: For a general case, how does the expected discounted return change if a constant offset is added to all rewards?*)

   Solution

   This is actually a trick question. It may appear that Option b is better (we are penalizing each extra move, so presumably we will reach the goal quicker using option b). However, there actually is no benefit of one over the other. This is because the rewards are uniformly shifted by a constant offset (of $\delta = -0.5$). Here is a formal proof. Recall the definition of the value function:

$$V(s) = \mathbb{E}\{G_t | s_t = s\}$$

   Now, in the first case, the discounted return is given by:

$$G_t^a = r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} \ldots = \sum_{i=0}^{\infty} \gamma^i r_{t+i},$$

   while in the second case, the discounted return is given by:

$$\begin{aligned} G_t^b &= r_t + \delta + \gamma(r_{t+1} + \delta) + \gamma^2(r_{t+2} + \delta) + \ldots \\ &= \sum_{i=0}^{\infty} \gamma^i r_{t+i} + \sum_{i=1}^{\infty} \gamma^i \delta \\ &= G_t^a + \frac{\delta\gamma}{1-\gamma}. \end{aligned}$$

   Hence, the value functions are the same in both cases, except with a constant offset of $\frac{\delta\gamma}{1-\gamma}$:

$$V^b(s) = V^a(s) + \frac{\delta\gamma}{1-\gamma}.$$

   Therefore, in both Case a and Case b, the Q-values of different state-action pairs remain the same *relative to each other*, and any gradient-based Q-learning procedure will lead to the same policy in the end. The intuition is the same as in regular supervised/unsupervised learning: it does not matter if you use a loss function L or add some constant shift to L everywhere; gradient descent will lead to the same weights.

3. **(4 points)** Open the (incomplete) Jupyter notebook provided as an attachment to this homework in Google Colab (or other environment of your choice) and complete the missing items. Save

your finished notebook in PDF format and upload along with your answers to the above theory questions in a single PDF.

Solution

A solution notebook has been posted on NYUClasses.