Spring 2021

Stanford
AA 203: Optimal and Learning-based Control
Problem Set 4, due June 4 by 5:00 pm

Problem 1: Deep reinforcement learning

As seen in lecture and in homework 3, a naïve Monte Carlo policy gradient method is quite unstable, due to its high variance. To address this, we will implement an actor-critic method, in which a value function is used as baseline (referred to as the critic, which predicts the cost associated with actions), with the policy referred to as the actor (which selects the actions). In particular, we will implement advantage-based actor-critic algorithm that was discussed in lecture 14. Combining the policy gradient Q-function with the value function baseline we obtain the following formulation:

$$\nabla_\theta J(\theta) = \mathbb{E}\left[\delta^\pi \nabla_\theta \log \pi(\mathbf{u}_\tau \mid \mathbf{x}_\tau)\right] \tag{1}$$

with: $\delta^\pi = \sum_{\tau=t}^{N} r_\tau - V^\pi(\mathbf{x}_t)$ and $\mathbb{E}[\delta^\pi \mid \mathbf{x}, \mathbf{u}] = A^\pi(\mathbf{x}, \mathbf{u})$ the advantage.

To implement this method in practice, we fit our value function (for the critic) to target values:

$$y_t = \sum_{\tau=t}^{N} r_\tau \tag{2}$$

and use gradient descent to regress onto these target value with the following objective:

$$\min_w \sum_{i,t} (V_w^\pi(\mathbf{x}_{it}) - y_{it})^2. \tag{3}$$

Implementation: We have provided starter code and added **TODO**s in the following functions that are required to be implemented:

- `forward`: Sets up the neural network
- `select-action`: Selects the action
- `finish-episode`: Training code, computes actor and critic loss

Results: We will implement the method on the `LunarLanderContinuous-v2` environment. To get good results, you should run the algorithm for approximately 1500-2000 episodes. Provide a plot of the total reward versus episode. Run the script as follows (render can be toggled to True or False (default) to view the performance of the agent):

```
python run_actor_critic.py --render True
```

Improvements (optional): Many improvements over the method implemented above are possible and can improve performance. A non-exhaustive list is given below, feel free to experiment with these (you can also try them out on more complicated environments such as `BipedalWalker-v2`):

- Experience replay: Stores transitions (x_t, u_t, r_t, x_{t+1}) in a buffer. Old examples are deleted as we store new transitions. To update the parameters of our network, we sample a (mini-)batch from the buffer and perform the stochastic gradient update on this batch.
- Bootstrap with value function, instead of using the sum of the tail rewards to estimate advantage.
- Iterate between computing target values and updating the value function to regress to these target values. You can add parameters that control the number of target value updates and the number of gradient updates.
- Using different policy or value networks.

Problem 2: Extremal curves

Given the functional

$$J(x) = \int_0^1 \left(\frac{1}{2}\dot{x}(t)^2 + 5x(t)\dot{x}(t) + x(t)^2 + 5x(t) \right) dt,$$

find an extremal curve $x^* : [0, 1] \to \mathbb{R}$ that satisfies $x^*(0) = 1$ and $x^*(1) = 3$.

Problem 3: Minimum control effort

Consider the dynamics

$$\dot{x}(t) = -2x(t) + u(t)$$

with the initial constraint $x(0) = 2$, terminal constraint $x(1) = 0$, and cost functional

$$J(u) = \int_0^1 u(t)^2 \, dt.$$

Write down the Hamiltonian and use the necessary optimality conditions to derive an optimal control $u^*(t)$ and corresponding state trajectory $x^*(t)$.

Problem 4: Zermelo's ship

Zermelo's ship must travel through a region of strong currents. The position of the ship is denoted by $(x(t), y(t)) \in \mathbb{R}^2$. The ship travels at a constant speed $v > 0$, yet its heading $\theta(t)$ can be controlled. The current moves in the positive x-direction with speed $w(y(t))$. The equations of motion for the ship are

$$\begin{aligned} \dot{x}(t) &= v \cos\theta(t) + w(y(t)) \\ \dot{y}(t) &= v \sin\theta(t) \end{aligned}.$$

We want to control the heading $\theta(t)$ such that the ship travels from a given initial position $(x(t_0), y(t_0)) = (x_0, y_0)$ to the origin $(0, 0)$ in minimum time.

a) Suppose $w(y(t)) = \frac{v}{h} y(t)$, where $h > 0$ is a known constant. Show that an optimal control law $\theta^*(t)$ must satisfy a linear tangent law of the form

$$\tan \theta^*(t) = \alpha - \frac{v}{h} t$$

for some constant $\alpha \in \mathbb{R}$.

b) Suppose $w(y(t)) \equiv \beta$ for some constant $\beta > 0$. Derive an expression for the optimal transfer time $t_1^* - t_0$.

Learning goals for this problem set:

Problem 1: Gain experience implementing a neural network based actor-critic algorithm, to give insight into modern reinforcement learning practice.

Problem 2: To familiarize with the process of solving calculus of variations problems.

Problem 3: To familiarize with necessary conditions for minimum control effort problems.

Problem 4: To familiarize with necessary conditions for minimum time problems.