CS 277 (W22): Control and Reinforcement Learning
Assignment 3

Due date: Tuesday, February 22, 2022 (Pacific Time)

Roy Fox
https://royf.org/crs/W22/CS277

In the following questions, a formal proof is not needed (unless specified otherwise). Instead, briefly explain informally the reasoning behind your answers.

Part 1 Properties of linear–Gaussian systems (30 points)

Question 1 (7 points) It follows from the Cayley–Hamilton theorem that, for an $n \times n$ matrix A and $k \geq n$, A^k can be expressed as a linear combination of $\{I, A, \ldots, A^{n-1}\}$. Show that this implies that, for any vector $x \in \mathbb{R}^n$ and $k \geq n$, if $A^n x$ can be expressed as a linear combination of the columns of the controllability matrix $\mathcal{C} = \begin{bmatrix} B & AB & \cdots & A^{n-1}B \end{bmatrix}$, then so can $A^k x$.

Question 2 (7 points) In a discrete-time linear time-invariant (LTI) system (A, B), we called a state x' *reachable* from a state x if there exists some finite time $t \geq 0$ and a control sequence $u_0, \ldots u_{t-1}$, such that $x_t = x'$ if $x_0 = x$. If x' is reachable from x, we also say that x is *controllable* to x'. Use the result in the previous question to show that all states $x \in \mathbb{R}$ are controllable to the origin $x' = 0$ (we call this *full controllability*) if and only if the columns of A^n are spanned by $\mathcal{C}$.

Question 3 (8 points) Consider a deterministic uncontrolled LTI system with dynamics $x_{t+1} = Ax_t$ that is partially observable with no observation noise, i.e. $y_t = Cx_t$. The *observability matrix* of the system (A, C) is

$$\mathcal{O} = \begin{bmatrix} C \\ CA \\ \vdots \\ CA^{n-1} \end{bmatrix}.$$

We say that a state $x \neq 0$ is *unobservable* if, assuming $x_0 = x$, we have $y_t = 0$ for all $t \geq 0$. Show that no states are unobservable (we call this *full observability*) if and only if $\mathcal{O}$ has full (column) rank n.

Question 4 (8 points) Show that a system (A, C) as in the previous question is fully observable if and only if we can uniquely find x_0 after seeing enough observations $y_0, \ldots, y_{t-1}$.
Hint: note that any full column rank matrix M has a left inverse $M^\dagger M = I$. For the converse, if $x_0 = x$ and $x_0 = x'$ induce the same observation sequence, which state is unobservable?

Part 2 Actor–Critic Policy Gradient (40 points)

In this part you'll implement an Actor–Critic Policy Gradient algorithm. In all coding questions, append a printout of your code as a page in your PDF.

Question 1 (10 points) Download the code at https://royf.org/crs/W22/CS277/A3/a2c.py. In the function `actor_critic_loss`, write TensorFlow code that calculates a loss with 3 terms:

- An actor loss: a policy-gradient loss with pre-computed advantage estimates (`advantages`);
- A critic loss: a temporal-difference loss, the square error between the pre-computed value targets (`value_targets`) and the critic values, weighted by `critic_loss_coeff`; and
- A negative-entropy loss on the actor policy, weighted by `entropy_loss_coeff` (i.e. a slight push to *maximize* entropy). First try without it, and then add it and compare. Hint: `action_dist.entropy()` can come in handy.

Question 2 (10 points) In the function `postprocess_advantages`, recall that `sample_batch` is part of a single trajectory, but in this assignment we will **not** assume that it's the entire episode. The batch contains tuples (s_t, a_t, r_t, s'_t) for some consecutive steps $t \in \{t_1, \dots, t_2\}$ in a trajectory.

Write code that calculates the scalar `last_value_pred`, defined as $V_\phi(s'_{t_2})$, i.e. the critic's prediction of the expected return following the `next_obs` s'_{t_2} at the end of the batch in the sample.

Useful: (a) `policy._value`, a function that gets an array of observations and returns a same-size array of value predictions; and (b) `dones`, a boolean array indicating episode termination in each time step (hint: why is this useful here?).

Question 3 (10 points) Write NumPy code that calculates for each step the discounted one-step value targets for the critic's TD-learning and the discounted one-step advantages for the actor's policy gradient.

Question 4 (10 points) Run your code on the `CartPole-v1` environment for 1000000 time steps and report the results.

Part 3 Generalized Advantage Estimation (30 points)

Recall the definition of the GAE[1] as

$$A^\lambda(s_t, a_t) = \sum_{\Delta t} (\lambda\gamma)^{\Delta t} A(s_{t+\Delta t}, a_{t+\Delta t}).$$

[1] https://arxiv.org/abs/1506.02438

Question 1 (5 points) Write down a mathematical expression for the advantage estimate $A^\lambda(s_t, a_t)$ using the rewards $r_t, r_{t+1}, \ldots$ and the value estimates $V_\phi(s_t), V_\phi(s_{t+1}), \ldots$.

Question 2 (10 points) Create a copy of `a2c.py` called `gae.py`, and change it to use A^λ as the advantage estimates. Append a printout of your code as a page in your PDF.

Useful: the helper function `ray.rllib.evaluation.postprocessing.discount_cumsum` can come in handy.

Question 3 (7 points) Run your code on `CartPole-v1` with a variety of λ values.

Tip: by setting the name of the `trainer` to include the value of λ, you can easily see it later in TensorBoard.

Visualize the results in TensorBoard, and attach the resulting plots.

Question 4 (8 points) Briefly discuss the results, including:

- What was the best value of λ in your experiments?
- What happens as $\lambda \to 0$?
- What happens as $\lambda \to 1$ in theory? What happens in practice?