CS 277 (W22): Control and Reinforcement Learning
Assignment 2

Due date: Tuesday, February 8, 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 Relation between BC and PG (20 points)

Question 1.1 (8 points) Suppose that we want to imitate an unknown expert π^*, but we only have access to a dataset $\mathcal{D}$ of demonstrations provided by a known non-expert policy π_0. Suppose that the exploration policy π_0 supports the actions of π^*; i.e., $\pi_0(a|s) > 0$ for any (s, a) with $\pi^*(a|s) > 0$. Suppose that a teacher provides importance weights $\rho(\xi) = \frac{\pi^*(\xi)}{\pi_0(\xi)}$ for each $\xi \in \mathcal{D}$.

We would like to use Behavior Cloning (BC) with a Negative Log-Likelihood (NLL) loss to train a policy $\pi_\theta(a|s)$ on data $\xi \sim \mathcal{D}$ to imitate the expert π^*. What is the loss $\mathcal{L}_\theta^{\text{BC}}(\xi)$ on which we should descend?

Question 1.2 (4 points) Compare the answer to the previous question to the REINFORCE algorithm. How are they similar? How are they different?

Question 1.3 (8 points) Suppose that, instead of the importance weights $\rho(\xi)$, the teacher labels each trajectory ξ with its return $R(\xi)$. Let $J_\theta = \mathbb{E}_{\xi \sim p_\theta}[R(\xi)]$ be the RL objective for policy π_θ. Write a loss $\mathcal{L}_\theta^{\text{PG}}(\xi)$ whose gradient with respect to θ, on data $\xi \sim \mathcal{D}$, is an unbiased estimate of $\nabla_\theta J_\theta$. Note that only π_θ, π_0, ξ, and $R(\xi)$ are available.

Part 2 Advantage estimators (30 points + 10 bonus)

You are playing an infinite sequence of Rock–Paper–Scissors rounds. After each round, you get a reward of 1, 0, or -1, respectively if you win, tie, or lose. You opponent always tries to beat your previous action (e.g. to play Paper if you previously played Rock) with probability 40%, and selects the other two actions with probability 30% each; initially, they play as if you've just played Paper. Knowing this, you play Rock, then Scissors, then Paper, and repeat. You estimate that, with discount factor 0.95, this policy gives you an expected future discounted return of $V(s) = 5$, for each state s (there is no variance in this estimate).

Question 2.1 **(5 points)** What is the expectation and variance of the reward $r(s, a)$ in each state s and action a?

Question 2.2 **(5 points)** In Policy-Gradient with an advantage estimator of the form $A(s_t, a_t) = Q(s_t, a_t) - V(s_t)$, why do we care about bias in Q but not in V?

Question 2.3 **(7 points)** Consider the Monte-Carlo advantage estimator

$$A^{\text{MC}}(s_t, a_t) = \sum_{\Delta t=0}^{\infty} \gamma^{\Delta t} r_{t+\Delta t} - V(s_t),$$

for each state s_t and action a_t, and for on-policy experience. Due to the previous question, we will analyze $Q^{\text{MC}}(s_t, a_t) = A^{\text{MC}}(s_t, a_t) + V(s_t)$. What is the bias and variance of this $Q^{\text{MC}}(s_t, a_t)$ estimator, for any state s_t and the optimal action a_t?

Question 2.4 **(7 points)** Consider the n-step advantage estimator

$$A^n(s_t, a_t) = \sum_{\Delta t=0}^{n-1} \gamma^{\Delta t} r_{t+\Delta t} + \gamma^n V(s_{t+n}) - V(s_t),$$

for each state s, action a, and $n \geq 1$, and for on-policy experience. What is the bias and variance of the corresponding $Q^n(s_t, a_t)$ estimator, for any state s_t and the optimal action a_t?

Question 2.5 **(6 points)** The mean square error (MSE) of an estimator is given by the sum of its variance and square bias. Which n minimizes the MSE of the n-step estimator $Q^n(s_t, a_t)$?

Question 2.6 **(5 bonus points)** Consider the GAE(λ) advantage estimator

$$A^\lambda(s_t, a_t) = (1 - \lambda) \sum_{n=1}^{\infty} \lambda^{n-1} A^n(s_t, a_t) = \sum_{\Delta t=0}^{\infty} (\lambda\gamma)^{\Delta t} A^1(s_{t+\Delta t}, a_{t+\Delta t}),$$

for each state s, action a, and $\lambda \in [0, 1]$, and for on-policy experience. What is the bias and variance of the corresponding $Q^\lambda(s_t, a_t)$ estimator, for any state s_t and the optimal action a_t?

Question 2.7 **(5 bonus points)** Which λ minimizes the MSE of the GAE(λ) estimator $Q^\lambda(s_t, a_t)$?

Part 3 Model-Free Reinforcement Learning algorithms (50 points)

Question 3.1 **Policy Gradient (5 points)** Download the code at https://royf.org/crs/W22/CS277/A2/pg.py.

In the function `policy_gradient_loss`, write TensorFlow code that computes the Policy Gradient loss. (Hint: arithmetic operators work for TF tensors, and TF has build-in functions for NumPy-like operators, e.g. reduce_sum.)

In the function `calculate_returns`, write NumPy code that computes the return of steps in `sample_batch`. `sample_batch` is guaranteed to represent part of a single trajectory, and in this assignment we'll assume it's the entire trajectory. The return will be the sum of rewards along the trajectory. You can discount the sum however you'd like, or not at all.

Note that we need `returns` to be a 1-D NumPy array of the same size as the rewards, i.e. the length of the trajectory. Create an array with the same return repeated in each element.

Run `pg.py`. Take note of the "Result logdir" that RLlib prints after each evaluation. When running the algorithm, you can specify a different logdir, or just use the default like we did here.

Behold your creation:

```
rllib rollout <logdir>/<checkpoint_num>/<checkpoint-num> --run PG
  --env CartPole-v1 --steps 2000
```

Append a printout of your code as a page in your PDF.

Question 3.2 Policy Gradient with Future Return (15 points) We can reduce the variance of the gradient estimator by not taking into account past rewards. Copy `pg.py` as `pg2.py`, and change `calculate_returns` to sum (with or without discounting, but be consistent with what you did before) only future rewards in each step. (Hint: the functions `numpy.cumsum` and `ray.rllib.evaluation.postprocessing.discount_cumsum` can come in handy, but be careful how you use them.)

Tip: Don't forget to change the `name` of the `PG_Trainer`.

Run `pg2.py` and view the results.

Append a printout of your code as a page in your PDF.

Question 3.3 DQN (10 points) Download the code at https://royf.org/crs/W22/CS277/A2/dqn.py. Run it and view the results.

```
rllib rollout <DQN logdir>/<checkpoint_num>/<checkpoint-num> --run DQN
  --env CartPole-v1 --config '{"dueling": false}' --steps 2000
```

`dones` is a vector of booleans that, for each time step, indicates whether the next state (reached at the end of the step) terminated the episode.

Explain the role of `dones` in line 43.

Question 3.4 Double DQN (15 points) Fix `dqn.py` such that, if `policy.config["double_q"]` is `True`, the loss will be the Double DQN loss:

$$\mathcal{L}_\theta(s, a, r, s') = (r + \gamma Q_{\bar{\theta}}(s', \operatorname*{argmax}_{a'} Q_\theta(s', a')) - Q_\theta(s, a))^2,$$

where $\bar{\theta}$ are the parameters of the target network.

Change `policy.config["double_q"]` to `True`, run your code, and view the results.

Append a printout of your code as a page in your PDF.

Question 3.5 Visualize results (5 points) TF comes with a utility for visualizing training results, called TensorBoard.

Run a TensorBoard web server:

```
tensorboard --logdir <the result logdir from the previous sections>
```

Take note of the URL in which TensorBoard is now serving (likely http://localhost:6006/). Open a browser at that URL. Take some time to make yourself familiar with the TensorBoard interface.

You should be able to see all the RLlib runs on the bottom left, with a color legend. If you happened to execute more runs than the four detailed above, uncheck all the other runs.

Find the plot tagged "tune/episode_reward_mean". You can find it manually, or use the "Filter tags" box at the top. Enlarge the plot using the left of 3 buttons at the bottom.

On the left you'll find some useful options. Uncheck "Ignore outliers in chart scaling" and note the effect on the plot.

Unfortunately, there's currently no good way to save the plot as an image, so just take a screenshot, and include it as a page in your PDF.

Question 3.6 Extra fun For extra fun, repeat the above experiments with other (discrete action space) environments from OpenAI Gym (https://gym.openai.com/envs), such as `Acrobot-v1`, `LunarLander-v2`, `Pong-v4`, and `Breakout-v4`. There's no extra credit, because getting good results will likely take more `--steps` and time than I should ask you to run.