CS 295: Optimal Control and Reinforcement Learning

Winter 2020

Assignment 3

due Tuesday, February 18 2020, 11pm

Part I

Consider the empirical estimation of the advantage of action a in state s, upon seeing (s, a, r, s'), as $\hat{A}(s, a) = r + \gamma V(s') - V(s)$. For the purpose of this question, we'll make the following assumptions:

- $V = V_\pi$ is the true state-value function of the policy of interest π;
- The experience (s, a, r, s') is sampled by rolling out π;
- $r(s, a)$ is a deterministic function of the state s and action a; and
- $V(s) \in [-1, 1]$ for all s.

1. Remind yourself that $\mathbb{E}_{a|s\sim\pi}[\hat{A}(s, a)] = 0$, by definition of V_π. What conditional distribution of $V(s') \in [-1, 1]$, given s and a, induces the worst-case conditional variance of $\hat{A}(s, a)$?

2. What is $\mathbb{E}[V(s')|s, a]$? What can you conclude about $r(s, a)$?

3. Suggest a two-state process (MDP and policy) that has the worst-case conditional variance of $\hat{A}(s, a)$ in each state and action. Specify $p(s'|s, a)$, $r(s, a)$, and $\pi(a|s)$ for each s, a, and s'. Don't worry if the process is very degenerate, in fact try to make it as simple as possible. Hint: originally, the policy was omitted in this question, because in the simplest worst-case MDP the policy doesn't make a difference.

4. In the process you proposed above, what is the conditional variance, given s_t and a_t, of each of the following advantage estimators on on-policy experience $(s_t, a_t, r_t, s_{t+1}, \ldots)$?

   (a) $\hat{A}^{\mathrm{MC}}(s_t, a_t) = \sum_{t' \geqslant t} \gamma^{t'-t} r_{t'} - V(s_t)$.

   (b) $\hat{A}^n(s_t, a_t) = \sum_{t'=t}^{t+n-1} \gamma^{t'-t} r_{t'} + \gamma^n V(s_{t+n}) - V(s_t)$; which n minimizes the conditional variance under the question's assumptions?

   (c) **Bonus:** $\hat{A}^\lambda(s_t, a_t) = (1-\lambda) \sum_{n \geqslant 1} \lambda^{n-1} \hat{A}^n(s_t, a_t) = \sum_{t' \geqslant t} \gamma^{t'-t} \hat{A}^1(s_{t'}, a_{t'})$; Hint: careful, these $\hat{A}$ aren't conditionally independent.

Part II

1 Advantage Actor–Critic

In this part you'll implement several actor–critic algorithms, starting with Advantage Actor–Critic (A2C; https://arxiv.org/abs/1602.01783). Note: in RLlib it's easy to distribute the algorithm's execution using Ray, by setting the `num_workers` in the Tune configuration. If the distribution is asynchronous (which is not the default), this would then be the A3C algorithm.

Download the code at https://royf.org/crs/W20/CS295/A3/a2c.py. In the function `actor_critic_loss`, write TensorFlow code that calculates a loss with 3 terms:

- A policy-gradient loss with advantage estimation;
- A temporal-difference loss for the value function, weighted by `v_loss_coeff`; and
- **Bonus:** a negative-entropy loss on the policy, weighted by `ent_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.

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 trajectory. Write code that calculates the scalar `last_value_pred`, i.e. the critic's prediction of the expected return following the last s' (a.k.a `next_obs`) in the sample. Useful: (1) `policy._value`, a function that gets an array of observations (a.k.a states, when observability is full) and returns a same-size array of value predictions (you can see below how it's implemented); and (2) `dones`, a boolean array indicating termination in each time step (hint: why is this useful here?).

Also write NumPy code that calculates the discounted one-step advantages and value targets.

Run your code on the `CartPole-v1` environment for 1000000 time steps.

2 Generalized Advantage Estimation

Create a copy of `a2c.py` called `gae.py`, and change it to implement the GAE algorithm (see https://arxiv.org/abs/1506.02438, also question 4c in Part I). Recall the helper function `ray.rllib.evaluation.postprocessing.discount`.

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. 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?