CS 277: Control and Reinforcement Learning (Winter 2021)
Assignment 4

Due date: Friday, February 26, 2021 (Pacific Time)

Roy Fox
https://royf.org/crs/W21/CS277/

General instructions: In theory questions, a formal proof is not needed (unless specified otherwise); instead, briefly explain informally the reasoning behind your answers. In practice questions, include a printout of your code as a page in your PDF, and a screenshot of TensorBoard learning curves (`episode_reward_mean`, unless specified otherwise) as another page.

Part 1 Model-based error accumulation (25 points + 10 bonus)

Consider a model-based reinforcement learning algorithm that estimates a model $\hat{p}$ of the true dynamics p, and then uses it for planning. In all parts of this question, we assume that we can plan optimally in the estimated model, with the true non-negative reward function.

1. Suppose that the estimated model is guaranteed to have

   $$\|p(s'|s,a) - \hat{p}(s'|s,a)\|_1 \leqslant \epsilon,$$

   for all s and a, and that the initial distribution $p(s_0)$ is known exactly.

   Show that $|\mathbb{E}_{p_\pi}[r_t] - \mathbb{E}_{\hat{p}_\pi}[r_t]| \leqslant \epsilon t r_{\max}$, for any policy $\pi(a|s)$. (10 points)

   Hint: show by induction that $\|p_\pi(s_t) - \hat{p}_\pi(s_t)\|_1 \leqslant \epsilon t$.

   Bonus: show the tighter bound $|\mathbb{E}_{p_\pi}[r_t] - \mathbb{E}_{\hat{p}_\pi}[r_t]| \leqslant \frac{1}{2}\epsilon t r_{\max}$. (10 points)

2. Conclude that planning with $\hat{p}$ is near-optimal: $\mathbb{E}_{p_\pi}[R] - \mathbb{E}_{p_{\hat{\pi}}}[R] \leqslant 2\frac{\gamma}{(1-\gamma)^2}\epsilon r_{\max}$ (or without the 2, given the bonus question above), where π is optimal for p and $\hat{\pi}$ is optimal for $\hat{p}$. Note that $\sum_t \gamma^t t = \frac{\gamma}{(1-\gamma)^2}$. (5 points)

3. Now suppose instead that the state space is continuous, and that both the true dynamics f and the model $\hat{f}$ are deterministic, with a known initial state s_0. Determinism implies that there exists an optimal open-loop policy, i.e. a sequence of actions.

   Suppose that the true dynamics, the model, and the reward function are all Lipschitz, i.e. there exists a constant L such that $\|f(s,a) - f(\hat{s},a)\|_2 \leqslant L\|s - \hat{s}\|_2$, for all s, $\hat{s}$, and a, and similarly for $\hat{f}$; and for r, i.e. $|r(s,a) - r(\hat{s},a)| \leqslant L\|s - \hat{s}\|_2$. Suppose that $L > 1$. Suppose further that the estimated model is guaranteed to have

   $$\|f(s,a) - \hat{f}(s,a)\|_2 \leqslant \epsilon,$$

   for all s and a.

   Let r_t and $\hat{r}_t$ be the rewards in step t when the same sequence of actions is taken in f and, respectively, in $\hat{f}$. Show that $|r_t - \hat{r}_t| \leqslant \frac{L^t-1}{L-1}L\epsilon$. (10 points)

Part 2 Finite-state controllers (25 points)

A finite-state controller (FSC) is a finite-state machine with a finite set $\mathcal{M}$ of memory states; an internal state update distribution which, upon observing o_t, updates from internal state m_{t-1} to m_t with probability $\pi(m_t|m_{t-1}, o_t)$; and an action distribution $\pi(a_t|m_t)$.

1. Given a FSC and POMDP dynamics $p(s_{t+1}|s_t, a_t)$ and $p(o_t|s_t)$, write down a forward recursion for computing the joint distribution of m_{t-1} and s_t; that is, show how to compute $p_\pi(m_t, s_{t+1})$ using p, π, and $p_\pi(m_{t-1}, s_t)$. Show how to recover from this joint distribution the predictive belief $p(s_t|m_{t-1})$. (10 points)

2. Given also a reward function $r(s_t, a_t)$, write down a backward recursion for evaluating $V_\pi(s_t, m_t)$; that is, show how to compute $V_\pi(s_t, m_t)$ using p, π, r, and $V_\pi(s_{t+1}, m_{t+1})$. (15 points)

Part 3 RNN policies (50 points)

1. In the `LunarLander` environment (`https://gym.openai.com/envs/LunarLander-v2/`), the observation is [x position, y position, x velocity, y velocity, angle, angular velocity, left leg contact (Boolean), right leg contact (Boolean)]. In the `Pong` environment (`https://gym.openai.com/envs/Pong-v0/`), the observation is the image that the Atari console would render to the screen (usually 84×84 pixels, after clipping, rescaling, and gray-scaling) . Alternatively, Atari environments are often "wrapped" to provide in every step the 4 most recent images, i.e. an observation shaped $4 \times 84 \times 84$ (this is called frame-stacking).

   In which of these 3 environments (`LunarLander`, `Pong`, and frame-stacked `Pong`) would you expect an agent to benefit the most and the least from having memory? (15 points)

2. Test your hypothesis. Use any algorithm implemented in RLlib (`https://docs.ray.io/en/latest/rllib-toc.html#algorithms`) with an RNN policy (set `use_lstm` to `True`) and with a memoryless policy. Report your results. (35 points)