IOC5262 Spring 2020: Reinforcement Learning **(Due: 2020/05/12, 23:55pm)**

Assignment 2: Function Approximation, Actor-Critic, TRPO, and CPO

Submission Guidelines: Please compress all your write-ups (photos/scanned copies are acceptable), report, and source code (along with the model snapshots) into one .zip file and submit the compressed file via E3.

Problem 1 (Trust Region Policy Optimization) (6+8+10+10+6=40 points)

In this problem, we will prove some key properties used in trust region policy optimization. The subproblem (a) is about the surrogate function, and the subproblems (b)-(e) are about the proof of the policy performance bound provided by (Achiam, ICML 2017). Let $\mathcal{S}$ be the state space, γ be the discount factor, μ be the initial state distribution, and d_μ^π be the discounted state visitation distribution. Also, under a policy π, let P^π be the transition matrix of the π-induced MRP.

(a) Recall from Page 25 of the slides of Lecture 12: The surrogate function $L_{\pi_\theta}(\pi_{\theta_1})$ is defined as

$$L_{\pi_{\theta_1}}(\pi_\theta) := \eta(\pi_{\theta_1}) + \sum_s d_\mu^{\pi_{\theta_1}}(s) \sum_a \pi_\theta(a|s) A^{\pi_{\theta_1}}(s,a). \tag{1}$$

Show that $L_{\pi_{\theta_1}}(\pi_\theta)$ satisfies $\nabla_\theta L_{\pi_{\theta_1}}(\pi_\theta)|_{\theta=\theta_1} = \nabla_\theta \eta(\pi_\theta)|_{\theta=\theta_1}$.

(b) Show that the following equation holds: for any function $f : \mathcal{S} \to \mathbb{R}$ and any policy π

$$(1-\gamma)\mathbb{E}_{s\sim\mu}[f(s)] + \mathbb{E}_{s\sim d_\mu^\pi, a\sim\pi, s'\sim P(\cdot|s,a)}[\gamma f(s')] - \mathbb{E}_{s\sim d_\mu^\pi}[f(s)] = 0. \tag{2}$$

Then, by the fact that the expected total discounted reward can be written as $\eta(\pi) = \frac{1}{1-\gamma}\mathbb{E}_{s\sim d_\mu^\pi, a\sim\pi(\cdot|s)}[R(s,a)]$, please also show that

$$\eta(\pi) = \mathbb{E}_{s\sim\mu}[f(s)] + \frac{1}{1-\gamma}\mathbb{E}_{s\sim d_\mu^\pi, a\sim\pi, s'\sim P(\cdot|s,a)}[R(s,a) + \gamma f(s') - f(s)]. \tag{3}$$

(c) Show that for any two policies π, π', we have $d_\mu^{\pi'} - d_\mu^\pi = \gamma(I - \gamma P^{\pi'})^{-1}(P^{\pi'} - P^\pi)d_\mu^\pi$. Subsequently, we have

$$\|d_\mu^{\pi'} - d_\mu^\pi\|_1 \le \frac{2\gamma}{1-\gamma}\mathbb{E}_{s\sim d_\mu^\pi}\big[D_{TV}(\pi'(\cdot|s)\|\pi(\cdot|s))\big], \tag{4}$$

where $\|\cdot\|_1$ and $D_{TV}(\cdot\|\cdot)$ denote the L_1-norm and the total variation divergence, respectively.

(d) Next, define an intermediate helper function:

$$Y_{\pi,f}(\pi') := \mathbb{E}_{s\sim d_\mu^\pi, a\sim\pi(\cdot|s), s'\sim P(\cdot|s,a)}\Big[\Big(\frac{\pi'(a|s)}{\pi(a|s)} - 1\Big)(R(s,a) + \gamma f(s') - f(s))\Big]. \tag{5}$$

Also, define $\varepsilon_f^{\pi'} := \max_s \Big|\mathbb{E}_{a\sim\pi'(\cdot|s), s'\sim P(\cdot|s,a)}[R(s,a) + \gamma f(s') - f(s)]\Big|$. Show the following lower bound on $\eta(\pi') - \eta(\pi)$:

$$\eta(\pi') - \eta(\pi) \ge \frac{1}{1-\gamma}\Big(Y_{\pi,f}(\pi') - 2\varepsilon_f^{\pi'} D_{TV}(d_\mu^{\pi'}\|d_\mu^\pi)\Big). \tag{6}$$

(e) By combining the results in (b)-(e), show the policy performance bound:

$$\eta(\pi') - \eta(\pi) \ge \frac{1}{1-\gamma}\mathbb{E}_{s\sim d_\mu^\pi, a\sim\pi'(\cdot|s)}\Big[A^\pi(s,a) - \frac{2\varepsilon^{\pi'}\gamma}{(1-\gamma)}\big(D_{TV}(\pi'(\cdot|s)\|\pi(\cdot|s))\big)\Big]. \tag{7}$$

Problem 2 (Approximately Solving CPO using Duality) (10 points)

Recall from Page 15 of the slides of Lecture 15: We would like to solve the optimization problem (OPT)

$$\begin{aligned} &\text{Minimize}_{\theta\in\mathbb{R}^d} && -g^T(\theta-\theta_0) && (8)\\ &\text{subject to} && c+B^T(\theta-\theta_0)\le 0 && (9)\\ & && \frac{1}{2}(\theta-\theta_k)^T H(\theta-\theta_k)-\delta\le 0. && (10)\end{aligned}$$

Based on the optimization theory, (OPT) is a convex optimization problem as both the objective and the constraints are convex functions. In this case, one standard way is to convert the constrained (OPT) into an unconstrained *dual* problem by defining the *dual function* $D(\lambda,\nu)$ as:

$$D(\lambda,\nu):=\min_{\theta\in\mathbb{R}^d}\Big\{-g^T(\theta-\theta_0)+\nu^T\big(c+B^T(\theta-\theta_0)\big)+\lambda\Big(\frac{1}{2}(\theta-\theta_k)^T H(\theta-\theta_k)-\delta\Big)\Big\}, \tag{11}$$

where λ and ν are called the *Lagrangian multipliers*. Moreover, we call the following the *dual problem* of (OPT):

$$\max_{\lambda\ge 0,\nu\ge 0} D(\lambda,\nu). \tag{12}$$

By the standard convex optimization theory, if there exists one strictly feasible point in (OPT), then the optimal value of the dual problem is the same as the original problem (usually called "strong duality").

In this problem, please show that the dual function $D(\lambda,\nu)$ of (OPT) can be written as:

$$D(\lambda,\nu)=\frac{-1}{2\lambda}(g^TH^{-1}g-2g^TH^{-1}B\nu+\nu^TB^TH^{-1}B\nu)+\nu^Tc-\frac{\lambda\delta}{2}. \tag{13}$$

Problem 3 (Policy Gradient Algorithms With Function Approximation) (30+20=50 points)

In this problem, we will implement two policy gradient algorithms with the help of neural function approximators: (1) REINFORCE with value function as the baseline; (2) Advantage Actor-Critic algorithm (A2C). For the pseudo code of the algorithms, please see the slides of Lecture 10. You may write your code in either PyTorch or TensorFlow (though the sample code presumes PyTorch framework). If you are a beginner in learning the deep learning framework, please refer to the following tutorials:

- PyTorch: https://pytorch.org/tutorials/
- Tensorflow: https://www.tensorflow.org/tutorials

For the deliverables, please submit the following:

- Technical report: Please summarize all your experimental results in 1 single report (and please be brief)
- All your source code
- Your well-trained models (REINFORCE and A2C) saved in either .pth files or .ckpt files

(a) We start by solving the simple "CartPole-v0" problem (https://gym.openai.com/envs/CartPole-v0/) using the REINFORCE algorithm with value function as the baseline function. Read through **reinforce_baseline.py** and then implement the member functions of the class **Policy** and the function **train**. Please briefly summarize your results in the report and document all the hyperparameters of your experiments.

(b) Based on the code for (a), implement the A2C algorithm by replacing the REINFORCE update with TD(0) bootstrapping and solve the "LunarLander-v2" problem, which has slightly higher state and action space dimensionality (https://gym.openai.com/envs/LunarLander-v2/). Save your code in another file named **a2c.py**. Please add comments to your code whenever needed for better readability. Again, please briefly summarize your results in the report and document all the hyperparameters of your experiments.