HW Assignment 9 (Due by 1:30pm on Sep 26)

1 Theory (30 points)

1. [**Q-Learning**, 10 points] Why is Q-Learning considered an off-policy RL algorithm?

2. [**Policy Iteration**, 20 points] Policy iteration uses the value function v_π of the previous deterministic policy π to derive a new policy π' as follows:

$$\begin{aligned}\pi'(s) &= \arg\max_a q_\pi(s,a) \\ &= \arg\max_a \mathbb{E}[R_{t+1} + \gamma v_\pi(S_{t+1})|S_t = s, A_t = a] \\ &= \arg\max_a \sum_{s',r} p(s',r|s,a)[r + \gamma v_\pi(s')]\end{aligned}$$

   Show that if $\pi'(s) = \pi(s)\ \forall s \in S$, then v_π satisfies the Bellman optimality equation, and thus π must be an optimal policy.

3. (*) [**Function Approximation**, 30 points] In class, the state-value function, $V(s)$ was represented using a lookup table. Correspondingly, the temporal difference (TD) update rule for SARSA was expressed as:

$$V(S_t) = V(S_t) + \alpha[R_{t+1} + \gamma V(S_{t+1}) - V(S_t)] \tag{1}$$

   where α is the learning rate and γ is the discount factor.

   However, it is possible to show that this lookup table is a special case of linear function approximation. Consider the parameterized value function:

$$V(\mathbf{x}_t;\theta) = \theta^T \mathbf{x_t} \tag{2}$$

   where $\mathbf{x}_t$ is a feature vector representing state S_t, and θ is a vector of parameters that is trained using the following update rule:

$$\theta = \theta + \alpha[R_{t+1} + \gamma V(\mathbf{x}_{t+1};\theta) - V(\mathbf{x}_t;\theta)]\mathbf{x_t} \tag{3}$$

   (a) The update in equation (3) can be seen as a gradient descent update that tries to minimize a cost function $J(\theta)$. Show the function $J(\theta)$ that is minimized and derive equation (3) as a gradient descent update.

   (b) Given a state space S with a finite number of states, show how to encode the states $s \in S$ into corresponding feature vectors $\mathbf{x}(s)$ such that the gradient update rule (3) becomes equivalent with the TD update rule (1), and thus $V(\mathbf{x}(s);\theta) = V(s)$, $\forall s \in S$.

2 Implementation (60 points)

This implementation exercise is adapted from the Berkeley Deep RL Class and the Deep RL Bootcamp held at Berkeley in August 2017.

In the first two problems, you will implement the two classic methods for solving Markov Decision Processes (MDPs) with finite state and action spaces:

- [**Problem 1**, 20 points]: Value Iteration (VI). *My implementation has 5 lines of code.*
- [**Problem 2**, 20 points]: Policy Iteration (PI). *My implementation has 7 + 5 lines.*

Both methods find the optimal policy in a finite number of iterations. The experiments here will use the Frozen Lake environment, a simple gridworld MDP that is taken from the `gym` package from OpenAI and slightly modified for this assignment. In this MDP, the agent must navigate from the start state to the goal state on a 4x4 grid, with stochastic transitions.

Both VP and PI require access to an MDP's dynamics model. This requirement can sometimes be restrictive – for example, if the environment is given as a blackbox physics simulator, then we won't be able to read off the whole transition model. In the third problem, you will implement Q-Learning, which can learn from this type of environments:

- [**Problem 3**, 20 points]: Sampling-based Tabular Q-Learning. *My implementation has 4 + 2 + 4 lines of code.*

In the experiments for this problem, you will learn to control a Crawler robot, using the environment already implemented in the `gym` package.

Implementation Details: For this assignment, you will need to use Jupyter Notebook. Instructions for installing the necessary packages and for activating the *deeprlbootcamp* environment are included in the `readme.md` file. Write code only in the "YOUR CODE HERE" sections in the 3 Notebook files indicated in bold. Skeleton code and more detailed instructions are provided in each notebook file.

ml4900/
- hw09/
  - `code/`
    - **Lab 1 - Problem 1.ipynb**
    - **Lab 1 - Problem 2.ipynb**
    - **Lab 1 - Problem 3.ipynb**
    - crawler_env.py
    - frozen_lake.py
    - misc.py
    - environment.yml
    - readme.md

3 Submission

Electronically submit on Blackboard a hw03.zip file that contains the hw09 folder in which you write code **only in the required files**.

On a Linux system, creating the archive can be done using the command:
`> zip -r hw03.zip hw03`.

Please observe the following when handing in homework:

1. Structure, indent, and format your code well.
2. Use adequate comments, both block and in-line to document your code.
3. Make sure your code runs correctly when used in the directory structure shown above.