Homework 3: Infinite Horizon Control & Reinforcement Learning

Collaboration in the sense of discussion is allowed, however, the work you turn in should be your own - you should not split parts of the assignments with other students and you should certainly not copy other students' solutions or code. ***If you discuss your homework with someone, please indicate their name(s) in your submission.*** *See the collaboration and academic integrity statement here:* https://natanaso.github.io/ece276b. *Books may be consulted but not copied from.*

Submission

You should submit the following files on **Gradescope** by the deadline shown at the top right corner.

1. Upload your solutions to problems in pdf format: **FirstnameLastname_HW3.pdf**. You may use latex, scanned handwritten notes (write legibly!), or any other method to prepare a pdf file. Do not just write the final result. Present your work in detail, explaining your approach at every step.
2. Upload all **python** code you have written for each problem in zip format: **FirstnameLastname_HW3.zip**. The zip file should also include a README file with a clear description of the files used for each problem and how they can be run.

Problems

1. [40 pts] Now that you have had 7 weeks of ECE276B, it is time to put it on the line. You take a flight to the Mohegan Sun Casino with your savings of $\$10,000$ in your bag. There will be some juicy games for Memorial Day. Indeed, the casino is offering a high-stakes modified roulette game with the following rules. If you bet on red, 70% of the time you win and double your bet and 30% of the time you lose your bet. If you bet on black, 20% of the time you win and you get back 11 times your bet (minus the 1 bet you invested, this is a profit of 10 bets) and 80% of the time you lose your bet. Your plan is simple – triple your money or go broke. More precisely, you have decided to stop playing if either you lose everything or as soon as your bankroll equals or exceeds $\$30,000$. You do not have time to waste so your plan is to bet in multiples of $\$10,000$. For example, initially the only bet you can make is $\$10,000$. Alternatively, if you had $\$20,000$, you could bet either $\$10,000$ or $\$20,000$.
   (a) Formulate your gambling objective as an infinite horizon stochastic shortest path problem with no discount ($\gamma = 1$). Your goal is, of course, to have the most money at the time you stop (either at \$0 or at $\geq \$30,000$). Clearly specify the state space, control space, motion model, stage cost, and terminal cost.
   (b) Given your formulation above, suppose that your control policy π is to play red in any non-terminal state. Use the policy evaluation theorem to approximate the value function V^π corresponding to the policy π. Compute a crude estimate $\bar{V}^\pi$ based only on a single iteration of the policy evaluation equation and a precise estimate $\hat{V}^\pi$ by either performing many iterations or solving the policy evaluation equation exactly.
   (c) Given $\bar{V}^\pi$ and $\hat{V}^\pi$, determine improved policies $\bar{\pi}'$ and $\hat{\pi}'$ using policy improvement. Are the policies the same and how do they compare to π?
   (d) Continue the steps above to compute an optimal policy π^* and the optimal value V^* using either Value Iteration or Policy Iteration. Explain how you should play the roulette game at Mohegan Sun based on π^* and how much you expect to walk away with. Is there going to be a celebration?
2. [40 pts] Consider the 5×5 environment shown in Fig. 1. The cells of the grid correspond to robot states. At each cell, four controls are possible: north, south, east, and west, which deterministically cause the robot to move one cell in the respective direction on the grid. Controls that would take the robot off the grid leave its position unchanged but result in a cost of 1. Other controls have zero cost, except those that move the robot out of the special states A and B shown in Fig. 1. From state A, all four controls yield a cost of -10 and take the robot to A′. From state B, all four controls yield a cost of -5 and take the robot to B′. Consider an infinite-horizon discounted problem aimed at minimizing the expected long-term cost for the robot using a discount factor of $\gamma = 0.9$.

(a) Compute the optimal value function and an optimal policy using value iteration.

(b) Compute the optimal value function and an optimal policy using policy iteration.

(c) Compute the optimal value function and an optimal policy using Q-value iteration.

(d) Change to $\gamma = 0.8$. Does this change the optimal solution? Change to $\gamma = 0.99$. What do you observe and why?

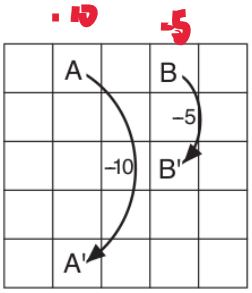

Figure 1: Grid environment with 25 states, two of which (A and B) have special dynamics.

3. [70 pts] Our goal is to solve the Hill Climbing problem (Fig. 2). Our automobile is stuck between two hills and its underpowered engine, even at full throttle, is unable to climb up the steep hill on the right. Our task is to find a control policy that takes advantage of the potential energy obtained by swinging between the two hills before driving straight towards the goal to the right. We will consider a simulation environment provided by OpenAI Gym (`https://gym.openai.com/`) and will compared the differences in on-policy and off-policy model-free planning algorithms. As the state space is continuous we need to apply a disretization to formulate a finite-state MDP with unknown motion model and cost function on which we will use TD policy iteration. We have provided a base class `HillClimbingProblem.py` which you can extend to implement the necessary functions. Keep in mind that you must determine a set of hyper-parameters that guarantees convergence to the optimal action-value function Q^*. For further information on the gym environments consider looking at the Gym Documentation.

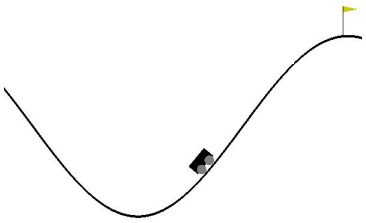

Figure 2: Hill Climbing Environment

We are aware that this is a well known problem and many ideas are available online. We will be very strict when it comes to plagiarism so please refrain from copying solutions. If you use references, please cite them!

(a) Determine a discretization of the state space that is suitable for planning.

(b) Implement an on-policy TD Policy Iteration algorithm (SARSA).

(c) Implement an off-policy TD Policy Iteration algorithm by using either importance sampling or Q-learning.

(d) Determine good hyper-parameters and compare the performance of the on-policy and off-policy TD PI algorithms

Write a project report describing your approach with the following sections:

- **Problem Formulation**: state the problem you are trying to solve in mathematical terms. Make sure to include the following:
  - MDP formulation with known and unknown variables,
  - optimization problem with proper objective function & constraints.

  In particular, you should include the basic elements of the model-free optimal control problem you are trying to solve.
- **Technical Approach**: describe your approach to TD evaluation and control and the key ideas that enable you to plan without a known model or cost function. Make sure to include,
  - explicit form of the algorithms,
  - necessary conditions for convergence to the optimal action-value function Q^*.
- **Results**: Present interesting details from your implementation and discuss the performance statistics of the two algorithms in detail under a set of hyper-parameters. Make sure to include plots of
  - $Q(x, u)$ over a number of episodes for a set of states,
  - the optimized policy $\pi(x)$ over the state space.

  What worked as expected and what did not and why? Consider including images or videos of the performance of your algorithms.

4. [100 pts] In this problem, our task is to solve an optimal control problem in order to balance an inverted pendulum. We will discretize the state and control spaces to formulate a finite-state MDP problem, and then solve it using value iteration and policy iteration. Consider the pendulum state $\mathbf{x} = (x_1, x_2)^T$ where x_1 is the angle of the pendulum and x_2 is the angular velocity. The continuous-time dynamics are:

$$d\mathbf{x} = f(\mathbf{x}, u)\, dt + \sigma\, d\omega \qquad f(\mathbf{x}, u) := \begin{bmatrix} x_2 \\ a \sin x_1 - b\, x_2 + u \end{bmatrix}$$

where $a > 0$ summarizes the effects of gravity, mass and length of the pendulum, $b > 0$ represents damping and friction, u is the control input and ω is Gaussian motion noise (Brownian motion). The stage-cost at each state $\mathbf{x}$ and control u is:

$$\ell(\mathbf{x}, u) = 1 - \exp(k \cos x_1 - k) + \frac{r}{2} u^2$$

where $k > 0$ determines the shape of the cost and $r > 0$ scales the control cost. Note that the cost penalizes x_1 that is away from 0 and is quadratic in the control u. Formulate an infinite-horizon Discounted problem with a discount factor $0 < \gamma < 1$. Experiment with the constants $(a, b, \sigma, k, r, \gamma)$.

Since the control and state spaces are continuous, we need to discretized them. First, you should decide a time step δt, maximum velocity v_{max} and maximum control u_{max}. You can discretize the state and control spaces into (n_1, n_2, n_u) number of grid points. Be careful that x_1 is an angle and should enforce wrap-around, while x_2 and u may be discretized over the intervals $[-v_{max}, v_{max}]$ and $[-u_{max}, u_{max}]$. Experiment with the constants $(\delta t, n_1, n_2, n_u, v_{max}, u_{max})$. For each discrete state $\mathbf{x}$ and each discrete control u, the transition probability $p_f(\mathbf{x}'|\mathbf{x}, u)$ is Gaussian with mean $\mathbf{x} + f(\mathbf{x}, u)\, \delta t$ and covariance $\sigma\sigma^T\, \delta t$. We have to create transition probabilities in the discretized MDP to approximate the Gaussian distribution defined by the continuous dynamics. We will choose possible next states/grid points around the mean, evaluate the Gaussian at the chosen grid points, and normalize so that the outgoing transition probabilities sum up to 1 at each state. The stage cost of the discretized MDP is $\ell(\mathbf{x}, u)\, \delta t$.

Generally, denser grids produce more accurate solutions. It is also useful to think about "compatibility" of the grid in the following sense. Suppose that the time step δt is small and the angular velocity discretization step is small, while the angle discretization step is large. Then, it will be very difficult for the MDP to make any state transitions, i.e., the transition probabilities will be close to delta functions centered at the current state. This is to be avoided. Instead, we should aim for

$$\frac{2 v_{max}}{n_2}\, \delta t \approx \frac{2\pi}{n_1}$$

and apply similar reasoning to the velocity and control discretization.

Now that we have formulated a discrete-space problem, we can use value iteration (VI) and policy iteration (PI) to obtain an optimal control strategy. For your choice of parameters, compare the convergence of VI and PI. After you solve the MDP, you have a control policy defined on the grid. Use interpolation to extend the control policy to continuous space and simulate several trajectories for the continuous system, starting at different initial states. Use `PendulumProblem.py` to help visualize your controller and if your controller (and problem and discretization parameters) are sensible. You should see the pendulum going to the vertical state and balancing there. Gradually increase the noise and discuss its effect. At what noise levels are you not able to achieve stable equilibrium in the vertical state?

Write a project report describing your approach with the following sections:

- **Problem Formulation**: state the problem you are trying to solve in mathematical terms. This section should be short and clear and should define the quantities you are interested in. In particular, you should include the basic elements of the optimal control problem you are trying to solve. Make sure to include:
  - the MDP formulation,
  - the optimization problem with proper objective function & constraints,
  - the interpolation problem
- **Technical Approach**: describe your approach to value & policy iteration and the key differences between these algorithms. Make sure to include:
  - the Value Iteration algorithm
  - the Policy Iteration algorithm
  - your policy interpolation method.
- **Results**: present any interesting details from your implementation and discuss the performance statistics of the two algorithms in detail. Make sure to include:
  - comparison of $V(x)$ over episodes between VI and PI for a set of states,
  - plots of the optimized policy $\pi(x)$ over the discretized state space for VI and PI.

  What worked as expected and what did not and why? Consider including images or videos of the performance of your algorithms.