Introduction to Artificial Intelligence (CS470): Assignment3

Deadline:

Sunday 23rd October, 2022

Setup

Option A: Google Colaboratory (Recommended)

Please, click on the following link to start the Colab session.

1. "File → Save a copy in Drive" to save it to your own drive. (This prevents "Warning: This notebook was not authored by Google.")
2. Once you have completed the assignment problem, you can save your edited files back to your Drive and move on to the next problem. Please ensure you are periodically saving your notebook: "File → Save" so that you don't lose your progress.

Option B: Local Development

You can also visit our GitHub repository, download an `assignment_3` folder, and start your own Colab session locally.

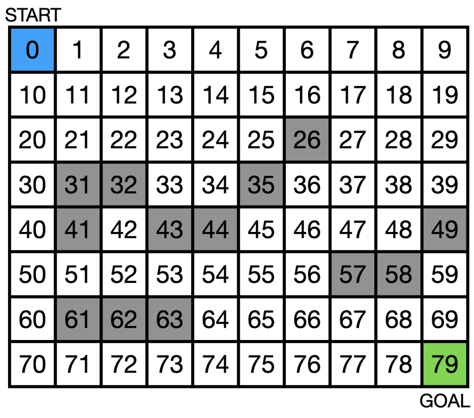
1. You need to install and launch jupyter notebook with following commands.
   ```
   pip3 install jupyter
   jupyter notebook
   ```
2. Turn your browser and connect to "http://localhost:8888/tree".
3. Run your session by clicking on `CS470_Assignment3_problem.ipynb` notebook file.

1 Markov Decision Process [50 pts]

In this section, you design a Markov decision process (MDP) for a toy environment, called *gridworld*, which is often used for reinforcement learning (RL). The environment is a stochastic version of the pre-built discrete *gridworld* environment from *OpenAI Gym*. In order to represent a task in the environment, you define an MDP, $< \mathcal{S}, \mathcal{A}, \mathcal{T}, r, \gamma >$, where $\mathcal{S}$, $\mathcal{A}$, $\mathcal{T}$, r ,and γ are states, actions, transition probabilities, a reward function, and a discount factor, respectively. We particularly use a 8×10 size of *gridworld* environment, where the coordinate of left-top and right-bottom cells are $[0, 0]$ and $[7, 9]$, respectively. An agent can move onto one of the four nearest cells or stay. Please, fill your code in the blank section following the "`PLACE YOUR CODE HERE`" comments in the `CS470_Assignment3_problem.ipynb` file following subproblems below.

TRANSITION MODEL: Implement a stochastic transition model of the environment dynamics. You need to fill out the `transition_model()` function, which returns a list of transition probabilities over the next states given a state and an action. In order to define the transition model, you consider following rules:

- The agent has five possible movements: stay, up, down, left, and right (see Fig.1),
- The agent is not allowed to move off the grid. When the agent tries to move off, it will end up staying at the previous location,



↻	Action 0	Stay
⬆	Action 1	Up
⬇	Action 2	Down
⬅	Action 3	Left
➡	Action 4	Right

Figure 1: An exemplar *gridworld* environment with available actions. An episode ends when the agent reaches either an obstacle (grey) cell or a goal (blue) cell.

- The agent moves once in a while to a (uniform) random location with a probability ϵ of failures,
- The agent terminates the episode when it reaches the goal or obstacle locations, and
- The agent cannot leave the goal or obstacle locations.

Note that we use predefined $\epsilon = 0.05$ on the assignment IPython notebook.

REWARD FUNCTION: Design a reward function by filling out the `compute_ reward(s,a,s')` function, where the arguments represent a current state, an action, and a next state. The function returns a positive reward *+10* when the agent reaches a goal, a negative reward *-5* when the agent reaches an obstacle, otherwise 0. In addition, the function provides a step penalty of -0.1 given any action.

STEP FUNCTION: Implement the `step()` function that takes an action and applies it to the environment. The applied action leads to a stochastic transition to the next state. In details, the `step` function performs followings:

- sample a next state according to the transition function,
- calculate a reward value,
- update the current state to the next state, and
- return a result tuple, (next state, reward, termination signal, information).

TERMINATION: We terminate an episode when an agent is in either goal or obstacle locations. To deal with the episode termination conditions, you have to implement the `is_done()` function that returns a Boolean termination signal that is whether the current episode has to be terminated or not.

Finally, you are ready to make your agent interact with the environment for RL. On your report,

1. **[15 pts]** print out the transition probabilities to all the next states given
   - a current state $[3, 3]$ and a selected action "Up",
   - a current state $[7, 4]$ and a selected action "Down",
   - a current state $[4, 9]$ and a selected action "Down",
2. **[5 pts]** plot the resulting histogram of returns produced by a dummy policy in the IPython notebook for 100 episodes,
3. **[5 pts]** plot the distribution of trajectories produced by a dummy policy in the IPython notebook for 300 episodes,
4. **[25 pts]** attach your implemented code from `transition_model()`, `compute_reward()`, `step()`, and `is_done()` functions.

2 Dynamic Programming [50 pts]

In this problem, you implement and analyze a representative DP algorithm: value iteration (VI).

2.1 Value Iteration (VI) [30 pts]

Implement the VI algorithm for the stochastic *gridworld* environment by filling out the `ValueIteration` class. The VI algorithm iteratively and directly updates each state value $V_t(s)$ at a time step t given a transition model $\mathcal{T}$:

$$V_{t+1}(s) = \max_a \mathbb{E}_{s' \sim \mathcal{T}(s,a)} \big[r(s,a,s') + \gamma V_t(s') \big]. \tag{1}$$

VI stops to update the values when the maximum update error Δ is lower than a certain threshold θ, where $\Delta \leftarrow \max(\Delta, \|V_{t+1}(s) - V_t(s)\|) \quad \forall s \in \mathcal{S}$. (See details on the RL book p.83.) On your report, please

- write down the state values of the first 10 states of the *gridworld* environment[1],
- overlay the best action at each state based on the state-action values,
- plot the distribution of trajectories produced by the trained policy for 100 episodes, and
- attach your implemented code from `value_iteration()` and `get_action()` functions on your report.

2.2 Comparison under different transition models [20 pts]

Suppose that the probability of taking a random action is 20% (i.e., $\epsilon = 0.2$). This transition model will affect the **exploration** process of VI. Thus, in this part, you are asked to compare/analyze the effects when you run VI with $\epsilon = 0.05$ and $\epsilon = 0.2$. On your report, please

- plot the expected returns per ϵ value with respect to the number of iterations until convergence (two graphs or one unified graph),
- overlaying the best actions at each state per ϵ value (two visualizations),
- plot the distribution of trajectories produced by the trained policy for 100 episodes per ϵ value (two visualizations), and
- compare/analyze the effect of different ϵ based on the above results.

Note that analysis does not mean you have to write down a long report. Scientific writing requires a concise delivery of your thoughts/results.

[1]The first 10 states are the top row of states: s_0, s_1, ... , s_9

Submission Guide

Submission Requirement

Change your file name to `cs470_yourname_studentID.ipynb`. Download and save on your machine. Write a report explaining how you implemented it, comparing the models, and discussing the test performance in a PDF file. Generate a zip file of your code and report, then save your zip file as `cs470_yourname_studentID.zip`. Please submit the `.zip` file via KLMS.

Please make sure that the submitted notebooks have been saved and the cell outputs are visible.

Academic Integrity Policy

This is homework for each student to do individually. Discussions with other students are encouraged, but you should write your own code and answers. Collaboration on code development is prohibited. There will be given no points in the following cases:

- Plagiarism detection
- Peer cheating
- The incompleteness of the code
- The code does not work