CS 4375 Introduction to Machine Learning
Fall 2021
Assignment 3: Markov Decision Processes and Reinforcement Learning

Part I: *Due electronically by Monday, October 11, 11:59 p.m.*

Part II: *Due electronically by Thursday, October 21, 11:59 p.m.*

Instructions:

1. Your solution to this assignment must be submitted via eLearning.
2. For the written problems, submit your solution as a **single PDF** file.
   - Only use **blue or black pen** (black is preferred). Scan your PDF using a **scanner** and upload it. Make sure your final PDF is **legible**. **Regrades due to non-compliance will receive a 30% score penalty**.
   - Verify that both your answers and procedure are **correct, ordered, clean, and self-explanatory** before writing. Please ask yourself the following questions before submitting:
     - Are my answers and procedure legible?
     - Are my answers and procedure in the same order as they were presented in the assignment? Do they follow the specified notation?
     - Are there any corrections or scratched out parts that reflect negatively on my work?
     - Can my work be easily understood by someone else? Did I properly define variables or functions that I am using? Can the different steps of my development of a problem be easily identified, followed, and understood by someone else? Are there any gaps in my development of the problem that need any sort of justification (be it calculations or a written explanation)? Is it clear how I arrived to each and every result in my procedure and final answers? Could someone describe my submission as messy?
3. **You may work individually or in a group of two**. Only one submission should be made per group. If you work in a group, **make sure to indicate both group members when submitting** through eLearning.
4. **IMPORTANT:** As long as you follow these guidelines, your submission should be in good shape; if not, we reserve the right to penalize answers and/or submissions as we see fit.

Part I: Programming (40 points)

In this problem you will implement the value iteration algorithm for finding the optimal policy for each state of an MDP using Bellman's equation. Your program should assume as input a file that contains a description of an MDP. Below is a sample input file:

```
s1 5 (a1 s1 0.509) (a1 s2 0.491) (a2 s1 0.31) (a2 s3 0.69)
s2 10 (a1 s1 0.4) (a1 s2 0.3) (a1 s3 0.3) (a2 s2 0.5) (a2 s3 0.5)
s3 -5 (a1 s1 0.3) (a1 s2 0.3) (a1 s3 0.4) (a2 s1 0.2) (a2 s2 0.8)
```

Each line in this file stores information for one state in the given MDP. For instance, the first line stores information about state `s1`: the reward associated with `s1` is 5, on action `a1` we stay in `s1` with probability 0.509 and move to `s2` with probability 0.491, and on action `a2` we stay in `s1` with probability 0.31 and move to `s3` with probability 0.69. The remaining lines of the file can be interpreted in a similar fashion.

After each of the first 20 iterations of the value iteration algorithm, your program should print to `stdout` the J value and the optimal policy for each state of the given MDP. Hence, the output of your program may look something like:

```
After iteration 1:  (s1 a1 5.0000) (s2 a1 10.0000) (s3 a1
-5.0000)
After iteration 2:  (s1 a1 11.7095) (s2 a1 13.1500) (s3 a2
3.1000)
After iteration 3:  (s1 a1 16.1751) (s2 a1 18.6029) (s3 a2
6.5757)
...
```

The first line of the above output says that after iteration 1, the optimal action in `s1` is `a1` and $J^1(\texttt{s1}) = 5$, the optimal action in `s2` is `a1` and $J^1(\texttt{s2}) = 10$, and the optimal action in `s3` is `a1` and $J^1(\texttt{s3}) = -5$. The remaining lines of output can be interpreted in a similar fashion.

Your program should allow exactly **four** arguments to be specified in the command line invocation of your program: (1) the number of states of the MDP, (2) the number of possible actions, (3) the input file as described above, and (4) the discount factor (γ). No other arguments are allowed. There should be no graphical user interface (GUI) of any kind. Any program that does not conform to the above specification will receive no credit. It may be helpful to take a look at the sample input and output files in the assignment page of the course website before you get started.

To implement the program, you should use Python, C++, or Java. Do **not** use any non-standard libraries in your code, and make sure your program can compile on UTD machines, not just your own machines. If you are using:

- **Python**
  - Make sure that your primary source file is `main.py` and that your code runs successfully after executing `python main.py <args>`.
- **C++**

    - Make sure that your primary source file is `main.cpp` and that your code runs successfully after executing `g++ main.cpp` and `./a.out <args>`.

- **Java**
    - Make sure that your primary source file is `Main.java` and that your code runs successfully after executing `javac Main.java` and `java Main <args>`.

If you have any questions, post them on piazza.

What to Submit

You should submit a single .zip file **via eLearning** (with your netID as zip file name), which includes your **source code** and your documentation. The names of all the members of the group should appear in the documentation accompanying your source code and in the comments box on the eLearning submission page. We will likely run your program on a new data set (with a different number of attributes) to test your code, so we encourage you to do the same!

Submission Structure and File Names

Your .zip file should have your netID as name e.g., gxo170730.zip. Inside the .zip there should be a single folder with your netID as name, which contains the corresponding files and folders. After extracting the .zip file, your submission should look as follows:

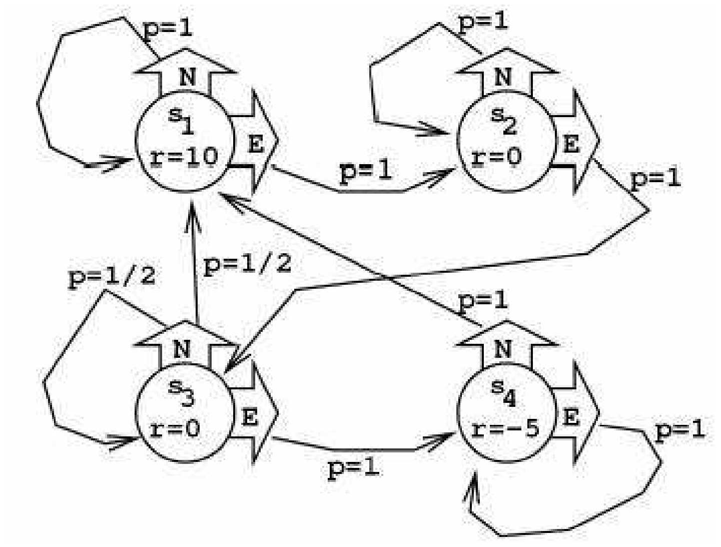
- /netID
    - /code
        - main.py

Note: You may include additional files as long as your code entry point and attachments are in the right location.

Part II: Written Problems (60 points)

1. **Policy Iteration (11 points)**

   Consider the following MDP:



   (a) **(4 pts)** Assuming the the initial policy $\pi_0(s)$ = N for all states s, compute J_0 and π_1. Use discount factor γ = 0.5.

   (b) **(4 pts)** Compute J_1 and π_2. Use discount factor γ = 0.5.

   (c) **(3 pts)** After computing which policy (i.e., which π_i) should you realize you have reached convergence?

2. **Value Iteration (11 points)**

   Consider again the MDP used in Problem 1.

   (a) **(4 pts)** Using value iteration, compute the J^* value for each state. You may use the program you wrote for Part I to compute these values. You do **not** need to show your work.

   (b) **(3 pts)** How many iterations of value iteration needs to be run in order to reach converagence, assuming that convergence is achieved if the maximum change in value between two iterations of any given state is less than 10^-4?

   (c) **(4 pts)** Using the values computed in part (a), derive the optimal policy. Show your work.

3. **Reinforcement Learning (36 points)**

   Consider training an MDP using the following sequence of states, actions, and rewards:

   S1, reward 0, action 1 $\rightarrow$
   S2, reward 1, action 1 $\rightarrow$
   S2, reward 1, action 2 $\rightarrow$
   S1, reward 0, action 1 $\rightarrow$

S2, reward 1, action 2 $\rightarrow$
S1, reward 0, action 2 $\rightarrow$
S3, reward 0, action 1 $\rightarrow$
S3, reward 0, action 1 $\rightarrow$
S4, reward 10, action 1 $\rightarrow$
S4, reward 10, action 2 $\rightarrow$
S4, reward 10.

Answer the questions below by using discount factor γ = 0.5 and learning rate α = 0.4.

(a) **(8 pts)** Use online supervised learning to calculate the J* value of each state. Show your work.

(b) **(8 pts)** Use temporal difference learning to calculate the J* value of each state. Show your work.

(c) **(8 pts)** Use certainty equivalent learning to calculate the J* value of each state. Show your work.

(d) **(12 pts)** Suppose you instead use Q-learning. Assume that all Q-values are initialized to 0. Fill in the table below to show how the Q-values change after the first six transitions.

State, Action Pair:	(S1, 1)	(S1, 2)	(S2, 1)	(S2, 2)	(S3, 1)	(S3, 2)	(S4, 1)	(S4, 2)
Q-value at start:	0	0	0	0	0	0	0	0
Q-value after observing: S1, reward 0, action 1 $\rightarrow$ S2								
Q-value after observing: S2, reward 1, action 1 $\rightarrow$ S2								
Q-value after observing: S2, reward 1, action 2 $\rightarrow$ S1								
Q-value after observing: S1, reward 0, action 1 $\rightarrow$ S2								
Q-value after observing: S2, reward 1, action 2 $\rightarrow$ S1								
Q-value after observing: S1, reward 0, action 2 $\rightarrow$ S3								