Assignment 4: Solving Markov Decision Processes
EECS 492: Artificial Intelligence
Fall 2015

Now: Thursday, November 19
Due: Tuesday, December 10, 11:00 pm

In this assignment, you will implement methods to solve a Markov Decision Process (MDP) for an optimal policy.

In the textbook [AIMA 3e], Markov Decision Processes are defined in Section 17.1, and Section 17.2 describes the Value Iteration approach to solving an MDP. Read these two sections very carefully. Section 17.3 describes the Policy Iteration approach, which is related, but is not used in this assignment.

If you have access to Sutton and Barto's *Reinforcement Learning* (1998), chapters 3 and 4 describe essentially the same material from a different perspective. Reading this would also be helpful, and helps you understand how it fits into the reinforcement learning picture.

There will be no provided code: you must write and debug your own code for this assignment, in either c++ or python. However if you need ideas, you may consult Java code from the AIMA code repository (http://aima.cs.berkeley.edu/code.html).

Assignment

1. Explore the impact of "cost of living" $R(s)$. This is based on Exercise 17.5 in the textbook.

   Implement an MDP model of the environment shown in Figure 17.1 (the discount factor γ is 1). Implement a solver that uses **value iteration** to find the optimal policy, given a value for $R(s) < 0$. Implement a binary search to identify the eight negative threshold values for $R(s)$ at which the optimal policy changes (the ninth threshold is zero). In the output, print only 4 decimal places of accuracy, i.e., your solution should be within ± 0.0001 of the correct value. Find the nine policies associated with the nine negative intervals between thresholds.

   > **Note 1**: Some of the values listed in Figure 17.2 and in the accompanying text may be incorrect in the least significant decimal places.
   >
   > **Note 2**: The states with +1 and -1 reward are terminal states. The utility of these states is the same as their reward, and should not be updated with value iteration.
   >
   > **Note 3**: Figure 17.4 contains pseudocode for value iteration. Since in our case, $\gamma = 1$, we can't use their convergence test. Instead, use $\delta < 0.0000001$ for convergence, where δ is the maximum change in utility. Value iteration converges rapidly, and this should take no more than 30 iterations.

   As an example, here's a display of the policy for $R(s) = -0.04$, laying the actions in a 4x3 array, corresponding to the states in the environment (use the character 'V' for a down action, and make sure to have spaces between all the states in a row).

   ```
   > > > 1
   ^ X ^ -1
   ^ < < <
   ```

Question 1 Why does the optimal policy for $R(s) = -0.04$ say to move left from state (3, 1), even though the utility for the state (3, 2) is higher than the state (2, 1) in Figure 17.3? Show your calculations.

2. In the same environment as problem 1, explore the difference between expected reward (average over many runs) and actual reward (on a particular run), given that the result of an action is uncertain.

   For the case of $R(s) = -0.04$, find and display the utilities for all the states to three decimal places. Implement a Monte Carlo simulation in which an agent starts at state (4, 1) and follows the optimal policy until it reaches a terminal state. Create three histograms of the final rewards earned by the agent for 10, 100, and 1000 runs, and compute the means and standard deviations of these runs. The final reward of a run is just the sum of the rewards of the states the agent is in at each time step.

Question 2 Compare these distributions with the expected utility of the initial state. Are the means similar to the expected utility, and to each other? How large a role does chance play? How spread out are the rewards in each distribution? Can you account for the shapes of the distributions?

3. Explore the impact of "discount rate" γ. This is an extension of Exercise 17.8 in the textbook.

   Implement an MDP model of the environment shown in Figure 17.14(a) and described in Exercise 17.8. Use a (non-terminal) reward of $r = +3$ for the upper left square. Use your value iteration solver to find the optimal policy, given a value for the discount rate γ. Implement a binary search to identify (out to five decimal places accuracy) five (5) threshold values for γ at which the optimal policy changes. Find the optimal policy for each of these six (6) intervals of γ values for $0 \leq \gamma < 1$.

Question 3 Intuitively, what does the discount rate mean, and why does changing it change the solution in this problem?

What to submit:

A zipped directory named "A4-*unique*", containing a solution pdf named *unique*-p4solution.pdf and a subdirectory named code. Replace "*unique*" with your own uniquename. For example, since my uniquename is rkruser, I would name the directory A4-rkruser and the pdf rkruser-p4solution.pdf.

The code subdirectory contains the source code for your program, which must compile and run in CAEN when the graders issue the following commands, after unzipping your file and changing to your code directory:

```
*** For C++***
g++ -std=c++11 -o main *.cpp
./main
*** For python ***
python main.py
```

Your program should take no arguments and automatically generate all of the following text files in a subdirectory of code named generated.

- For problem 1, generate a file P1-output.txt that contains the threshold values for $R(s)$, each on its own line, starting with zero (0), and followed in descending order by the eight (8) negative threshold values you have found. Between each pair of values, display the 4x3 array describing the optimal policy for that interval. Put an asterisk (*) after any action that changed from the previous policy.
- For problem 2, generate the file P2-output.txt, which contains the utilities for each state, in a 3×4 grid like the one you used to print the optimal policies (print 0 for the utility of the obstacle at (2,2)). Also generate 3 text files containing data for making the histograms, formatted as lists of numbers separated by newlines: P2-data-10.txt, P2-data-100.txt, and P2-data-1000.txt.
- For problem 3, generate a text file P3-output.txt with the threshold values for the discount rate, γ, separating each pair of adjacent threshold values with a 3x3 text display of the optimal policy for that interval, and a 3x3 text display of the utilities of the states in that interval (computed with γ equal to the lower threshold defining the interval), accurate to 3 decimal places.

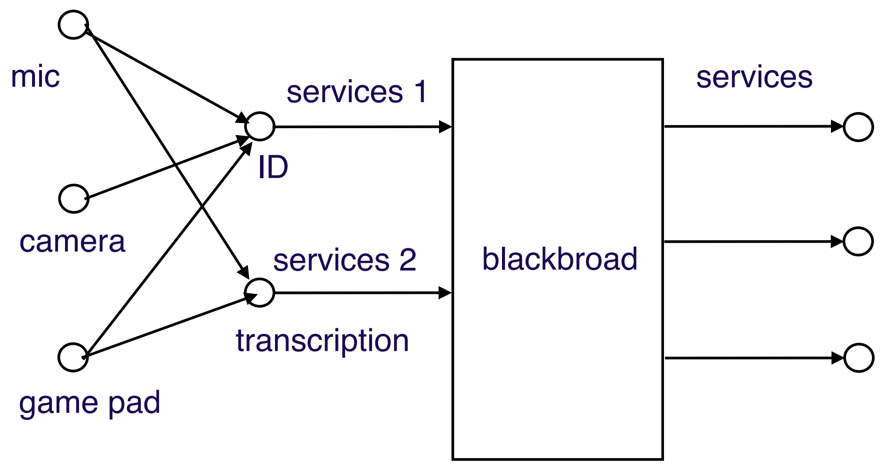
Your pdf should be formatted as follows:

- Page 1: Your name, and the answer to question 1.
- Page 2: The answer to question 2. Include in your answer the means and standard deviations of the histograms.
- Page 3: The answer to question 3.
- Page 4: The full output from P1-output.txt.
- Pages 5 and 6: The full output from P2-output.txt, and the three histograms you created. (If this only takes 1 page, leave page 6 blank).
- Pages 7: The full output from P3-output.txt.

The reason we ask for this format is that it is convenient to use with gradescope. Please stick to the given page numbers. Also, label each different item on pages 4-7 with a figure number and a title (for example, the output to P1-output.txt could be labelled "Figure 1: Problem 1 Output").

Grading

- 10 points for correct files and directory structure, and for code compiling and running without error in CAEN.
- Problem 1: 20 points for correct $R(s)$ thresholds and policies in P1-output.txt (about two points per threshold/policy). 10 points for answering question 1.
- Problem 2: 15 points for the histograms (5 for each) and 5 points for the correct table of utilities in P2-output.txt. 10 points for answering all parts of question 2.
- Problem 3: 20 points for the correct γ thresholds, policies, and utilities in P3-output.txt. 10 points for answering question 3.


mic
camera
game pad
services 1
ID
services 2
transcription
blackbroad
services