ASSIGNMENT #3 – COMP 4106 ARTIFICIAL INTELLIGENCE

The assignment is an opportunity to demonstrate and solidify your knowledge on Bayes theorem and rule-based systems.

The assignment may be completed individually, or it may be completed in small groups of two or three students. The expectations will not depend on group size (i.e. same expectations for all group sizes).

Components

The assignment should contain two components: an implementation and a technical document.

Implementation

The implementation should provide a complete solution to the problem outlined below.

Technical Document

Your technical document should outline the algorithms you used and answer any questions posed below. After reading the technical document, the reader should understand exactly how your implementation works.

The technical document will be graded both on content and on presentation quality.

If the assignment was completed in a small group of students, the technical document must include a statement of contributions. This statement should identify: (1) whether each group member made significant contribution, (2) whether each group member made an approximately equal contribution, and (3) exactly which aspects of the assignment each group member contributed to.

Logistics

Assignment due date: Monday, April 12, 2021

Assignments are to be submitted electronically through cuLearn. If you work in a small group, it is sufficient for one group member to submit.

It is your responsibility to ensure that your assignment is submitted properly. Copying of assignments is NOT allowed. Discussion of assignment work with others is acceptable but each individual or small group are expected to do the work themselves.

Implementation

Programming language: Python 3

You may use any standard Python libraries. You may also use the NumPy, Pandas, and SciPy packages. Use of any additional packages requires approval of the instructor.

You must implement your code yourself. Do not copy-and-paste code from other sources, but you may use any pseudo-code we wrote in class as a basis for your implementation. Your implementation must follow the outlined specifications. Implementations which do not follow the specifications may receive a grade of zero. Please make sure your code is readable, as it will also be assessed for correctness. You do not need to prove correctness of your implementation.

You may be provided with a set of examples to test your implementation. Note that the provided examples do not necessarily represent a complete set of test cases. Your implementation may be evaluated on a different set of test cases.

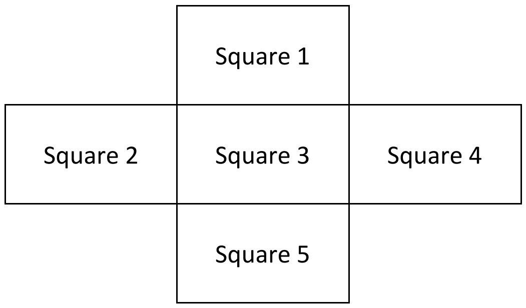
Technical Document

Page limit for technical document: 10 pages

Please use a standard page format (i.e. page size 8.5” x 11”, minimum ½” margins, minimum font size 10, no condensed fonts or spacing). The page limit includes all figures, tables, appendices, and references. Submit the technical document as a PDF file.

Implementation

Consider a vacuum cleaner agent working in a room with five squares, as illustrated below. In this assignment, we will use temporal difference Q-learning to learn the optimal policy for the vacuum cleaner.



Assume that the vacuum cleaner starts in the Square 1. It can take the following actions: clean the current square, move to a horizontally or vertically adjacent square.

Each square can either have the state dirty or clean (this is a binary state; there is no degree of dirtiness).

Assume that this is a fully observable environment. That is, the vacuum cleaner knows for all times its position and whether each square is clean or dirty.

Assume that this is a discrete time environment. At each time, the agent may take one action. Furthermore, at each time, with some probability, one square's state may change from clean to dirty.

Use the following string representations for states:
$PS_1S_2S_3S_4S_5$
Where P is the square number the vacuum cleaner is in
S_i = 0 if the Square i is clean; 1 if the Square i is dirty

Use the following string representations for actions:
"C": clean current square
"L": move left
"R": move right
"U": move up
"D": move down

The reward r associated with a state s is:
r(s) = -1 * number of dirty squares

Use the following parameters:
Gamma = 0.5 (discount factor)
Alpha = 0.1 (learning rate)

Initially estimate the Q-function as:
Q(s, a) = 0

A few important notes for your implementation:
1. Consider a single iteration of temporal difference Q-learning.
2. Use the provided simulated trajectories (which were generated under a random policy); do not generate new trajectories.

Your implementation must contain a file named “assignment3.py” with one class. The class must be named “td_qlearning”.

The “td_qlearning” class should have three member functions: “__init__”, “qvalue”, and “policy”.

The function “__init__” is a constructor that should take one input argument. The input argument is the full path to a CSV file containing a single trajectory through the state space. The CSV file will contain two columns, the first with a string representation of the sate and the second with a string representation of the action taken in that state. The ith row of the CSV file indicates the state-action pair at time i.

The function “qvalue” should take two input arguments. The first input argument is a string representation of a state. The second input argument is the string representation of an action. The function should output the Q-value associated with that state-action pair (according to the Q-function learned from the trajectory in the file passed to the __init__ function).

The function “policy” should take one input argument. The input argument is a string representation of a state. The function should output the optimal action (according to the Q-function learned from the trajectory in the file passed to the __init__ function).

The “qvalue” and “policy” functions will be called after the constructor is called. They may be called multiple times and in any order. I recommend computing the Q-function within the “__init__” function (store it in a member variable) and implement the “qvalue” and “policy” functions as getters.

Attached are example inputs and corresponding example outputs. Note that your functions should not write anything to file. These examples are provided in separate files for convenience.

Example trajectory CSV file:
```
101101, D
301101, C
301001, L
201011, C
200011, R
300011, R
400011, C
400101, L
300101, C
300001, D
500001, C
500000, C
500000, C
```

Example input to “qvalue” function:
```
201011
C
```

Example output from “qvalue” function:
```
-0.3
```

Example input to “policy” function:
```
201011
```

Example output from policy function:

R

Attached is skeleton code indicating the format your implementation should take. Code for testing your implementation will be provided.

Grading

The implementation will be worth 60 marks.

40 marks will be allocated to correctness on a series of test cases, with consideration to both the Q-function and the policy. These test cases will be run automatically by calling your implementation from another Python script. Ensure your implementation adheres exactly to the specifications.

20 marks will be allocated to human-based review of code.

Technical Document

The technical document should describe exactly how your implementation works.

In addition, please also answer the following questions in the technical document. Explain why your answers are correct.

1. What type of agent have you implemented?
2. Suppose there is a state-action pair that is never encountered during a trajectory through state space. What should the Q-value be for this state-action pair?
3. For some cases (even with a long trajectory through state space), the optimal Q-value for a particular state-action pair will not be found. Explain why this might be the case.
4. In the test cases provided, the trajectories through state space were generated using a random policy. Describe a different strategy to generate trajectories and compare it to using a random policy.

Grading

The technical document will be worth 40 marks.

20 marks will be allocated to the description of the implementation.

20 marks will be allocated to the questions posed above, with equal weight given to each question (i.e. five marks each).