CIS 421 Artificial Intelligence

Fall 2016 **Dr. Laura Grabowski**

Program 5: Q-learning – Trolls and Burglar

DUE DATES:
Phase 1 – Environment, agent movement, and reward structure: Monday, November 21, 11:00 PM
Phase 2 – Q-learning implementation and results: Wednesday, December 7, 11:00 PM

PROBLEM

Your agent is an intrepid burglar who has been sent to rescue your company's ponies from a group of trolls. Your task has three parts: rescue as many ponies as possible before the trolls eat them, avoid capture yourself, and escape.

PROGRAM

You will implement the Q-learning algorithm for your agent to learn the optimal policy for completing the task described above. Depending on what you implement, you will be graded according to different maximum possible credit for the program (see *Variations* below for details). The assignment will be submitted in two phases, to help you get started early.

All programs must include the following:

- Implementation of Q-learning for your burglar agent.
- Output: (1) visualization of (a) the initial environment, and (b) the path the agent follows using its learned policy, (2) the total reward received by the agent when performing the learned policy, (3) the ratio or percentage of ponies rescued by the agent in that environment (*e.g.,* if the agent rescues 4 of 6 ponies, you will report 4/6. 2/3, 66%, or 0.66; any of those forms is fine), and (4) the number of epochs needed to learn the policy. **Please write your program output to BOTH display AND to an output file.** You will have to include code to track the necessary information for this output.
- Program must accept input files for environments OTHER THAN the one that I provide for you. I must be able to enter the file name for the environment file from the command line at runtime. You will lose credit if I have to modify and recompile source code to change the input file. If the input requires a particular form, you must specify that in user prompts. AS always, you must include a README file that explains how to compile and run your program.

Environment

You will read the details of the environment from an input file. Environments will be N x N grids, with the possibility of internal obstructions. I will provide you with one environment, but your program may be graded using different environments (up to $N = 20$). Input files will have the following format:

- **Line 1:** value of *N*, number of trolls (1 to 3), number of ponies (1 to 15). This line will be 3 integer values separated by spaces.
- **Line 2:** Escape location (*x, y*) (for example, 8 3).
- **Line 3:** A series of (*x, y*) coordinate values defining the locations of each pony (all values separated by spaces; the first value in each pair is the *x* coordinate, the second is the *y* coordinate). As in the Vacuum Cleaner World of Program 1, coordinates are given as Cartesian coordinates, not as screen coordinates.

- **Line 4:** A series of (*x, y*) coordinate values defining the locations of each obstruction (all values separated by spaces; the first value in each pair is the *x* coordinate, the second is the *y* coordinate). If no interior obstructions exist, this line will be given as `-1 -1`.
- **Line 5:** A series of (*x, y*) coordinate values defining the locations of each troll (all values separated by spaces; the first value in each pair is the *x* coordinate, the second is the *y* coordinate).
- The burglar's initial location will be randomly generated at the start of each epoch, so the input file does NOT give location information for the burglar.

An environment will be visualized like this:[1]

```
## ## ## ## ## ## ## ## ## ## ##
## - -  -  -  -  -  -  -  - - ##
## - -  -  -  -  -  -  -  - - ##
## - -  -  T  P  -  -  -  - - ##
## - -  -  -  -  -  -  -  E - ##
## - -  -  -  P  -  -  -  - - ##
## - -  -  -  -  P  P  -  P - ##
## - -  -  P  -  -  -  -  - - ##
## - B  -  -  -  -  -  -  - - ##
## - -  -  -  -  -  -  -  - - ##
## - -  -  -  -  -  -  -  - - ##
## ## ## ## ## ## ## ## ## ## ##
```

where:

- `N` is 10 in this environment. Note that the edges of the grid are defined by ##, and lie OUTSIDE the edge of the world.
- `E` marks the spot where the agent can escape (i.e., the goal location).
- `P` shows the location of a pony.
- `T` shows the location of a troll.
- `B` shows the location of the burglar. For your first output grid, this denotes the initial location of the burglar; for the second output grid, you will show the final location of the burglar and the burglar's path from start to finish (using another symbol, such as 'X').
- -1 indicates locations where the burglar cannot move. Your code should handle the -1 as a signal that the agent cannot transition to those locations *(i.e.,* attempting to move into a cell with -1 should not be a valid action in the state).
- Locations that are empty are shown with `-`.

Important note: You should be able to use the drawing routine from the textbook code with relatively few modifications. You can also use the same format as the output for Vacuum Cleaner World (Reflex agent), again with few modifications.

[1] You may improve on this output if you like, or use the format from the Vacuum Cleaner World. The given output is the minimum, and is based on the output of the textbook code. Prettier output will NOT earn extra points.

Rewards and Parameters

Table 1. Values of rewards and parameters for basic program.

Rewards		Parameters	
Escape	15	Alpha (learning rate, called Beta in the textbook)	0.75
Pony	10	Gamma (discount factor)	0.5
Troll	-15	Epochs (max number of learning iterations)	10,000
Other locations	2		

Action Selection
The textbook code provides two action selection methods: greedy, for when you are exploiting (*i.e.*, using the learned policy) and probabilistic greedy (for when you are learning). You must include the greedy selection so that your learned policy can be tested. You may use either the probabilistic greedy method as in the book's code, or ε-greedy (with ε of ~ 0.1). If you feel ambitious, you may use Boltzmann exploration (see slide 31 of the RL lecture, "Softmax Action Selection"). To use this approach, you will need to have a "temperature" function similar to simulated annealing. Come talk to me if you want to try this action selection method.

GENERAL MOVEMENT RULES

- The agent can move in any valid direction (*i.e.,* any direction that doesn't have -1 at that location) using a Moore neighborhood (all surrounding locations **including** diagonals). For more information, see the explanation in the textbook pp. 214-215.
- Agents may share cells, except that trolls may NOT share cells with each other.
- Trolls eat other agents if the troll moves into a square occupied by another agent.
- The burglar rescues a pony when he moves into the cell occupied by the pony. Once a pony is rescued, you must remove the pony from the cell (don't forget to update the visualization). The only ponies that appear on the ending grid are ponies that the burglar did not rescue. The reward received for a rescued pony needs to be reflected in the burglar's total reward earned.

PHASES AND VARIATIONS
Your grade will be determined based on 2 submission phases and how many of the following variations you complete. **YOU MUST COMPLETE ALL PREVIOUS VARIATIONS TO BE SCORED ON A LATER ONE.** That means that you may not simply jump to variation 3 or 4 without completing 1 and 2. **If you do not follow this requirement, I will grade your program as if you did only variation 1.** You will report your results for EACH part of the assignment (in your output file; set it up to append and annotate your output file with the variation number).

Scoring Summary
Phase 1: 45 points
Phase 2: 55 points + 20 points extra credit

- **Variation 1 – basic Q-learning program. +25 points.**
- **Variation 2 – different learning rates and discount factors. +15 points.**
- **Variation 3 – variable learning rate, +15 points.**
- **Extra credit – moving trolls, +20 points.**

Phase 1, 45 points maximum.

For Phase 1, you must complete the following:

- Build the environment based on an input file as described above.
- Correctly identify the reward for each world state.
- Visualize the environment as described above.
- Correctly move the agent within the environment and show the path taken with a particular symbol (*e.g.,* ‘X’ or ‘o’).

If you do not submit your program for the Phase 1 deadline, your grade will automatically be penalized the full value of Phase 1 (45 points).

Phase 2, up to 120 points with extra credit

Variations

1. **Variation 1: Basic program. +25 points maximum.** Implement your program using static trolls. They are dangerous only if you move into the locations they occupy. Use the parameter values as given in Table 1 above. You will report only one set of results with this option.

2. **Variation 2: Test with different learning rates and discount factors. +15 points maximum.** In addition to the parameter values listed in Table 1, rerun your program using the sets of parameter values as shown in Table 2.

   Table 2. Parameter combinations for additional program runs.

Alpha (learning rate)	Gamma (discount factor)
0.1	0.5
0.5	
0.9	
0.6	0.1
	0.5
	0.9

   Report all the results as described above for required output. Along with the original parameter values (Table 1), this process will make for a total of **7 sets of results**. Be sure to include annotations for the parameter values that go with each group of results. In addition, include a brief analysis (2-4 paragraphs)[2] that discusses the effects of the different parameter values on learning. Things you might want to discuss include: Which combination of parameters works best? What happens with higher learning rate? What happens with higher discount factor? Do different parameters produce different behavior in the end? How do the parameter values affect the rate of convergence of the algorithm?

[2] You may write at the end of your output file, or include an additional word-processed file with your discussion.

3. **Variation 3: Variable learning rate. +15 points maximum.** In addition to testing and reporting the different fixed parameter values (Tables 1 and 2), adjust your program to implement a variable learning rate that is a function of the number of times a state has been visited. In general, the learning rate will be lower the more often a state has been visited.

4. **Extra credit: Trolls that move. +20 points.** Implement trolls that move with some probability at every time step. Trolls are quite stupid (and they drink a lot), so their movement will be random. Movement choices for trolls should include moving in any valid direction and remaining still. You may use any method for selecting movement, but be sure to include a description of your method in your program comments. The trolls will move before the burglar agent makes an action selection. You should be able to base your moving troll code on the existing movement support code from the textbook. Be sure to keep track of the changes in the environment.

WHAT YOU WILL TURN IN

The items that you submit will depend on how far you get through the variations of the program. Here is a summary by phase.

Phase 1:

1. Program code.
2. Input file.
3. README file with instructions on how to compile and run your program.

Phase 2:

1. Program code.
2. Input file.
3. Output file with all results (if you completed at least Variation 2).
4. README file with instructions on how to compile and run your program. Please also indicate which variations you completed.
5. Discussion (.pdf only please) if you got through Variation 2.