Artificial Intelligence (H) Assessed Exercise Report

Ruairidh Macgregor 2250079m

March 15, 2019

1   Introduction

The aim of this investigation was to evaluate the performance of three different agents in a computer simulated environment in which the agent has to navigate successfully through a frosty lake to reach the goal without falling in a hole (reaching a bad state). The three type of agents used were: a senseless/random agent that makes non-deterministic decisions at each state, a simple agent that has informed knowledge of the problem space so can use efficient searching algorithms to navigate to the goal and an reinforcement learning agent that learns an (potentially optimal) policy after running a number of episodes in the environment.

1.0.1   PEAS Analysis

A PEAS analysis on all three agents is as follows.

- **Performance**

  - Navigate to the goal state without reaching any bad states and in as few steps as possible.

- **Environment**

  - Frozen lake. Each state in the environment can either be a frozen surface (safe) or a hole (bad). The properties of the environment are as follows:
    * **Partially observable** for Senseless and Reinforcement Learning Agents and **Fully Observable** for the Simple Agent
    * **Single agent** in all cases
    * **Deterministic** for the simple agent and **stochastic** for the senseless and reinforcement learning agents.
    * **Sequential** in all cases
    * **Static Environment** in all cases

* **Discrete** in all cases
* **Unknown** for the senseless and reinforcement learning agents. **Known** for the simple agent.

- **Actuators**

  - Physical Movement in all cases

- **Sensors**

  - **None** for the senseless agent, **oracle** for the simple agent and **perfect information about the current state** for the reinforcement learning agent.

2 Method and Design

The environment itself was made using the Open-AI Gym[1]. The environment was instantiated with a given problem ID n ($n \in \{0..7\}, n \in \mathbb{Z}$), which affects the difficulty of the problem and the starting and goal states; a Boolean flag indicating if the environment was stochastic or not and the reward an agent will receive if the agent falls in a hole. If any of the three agents reach a hole, then the current trial is ended, the environment is reset and the agent restarts the trial. The AIMA Toolbox[2] was also used, more specifically for the algorithmic side of the investigation. For each agent, the agent executed a number of episodes of the problem and in each episode, the agent was allowed a maximum number of iterations. The NumPy[4], SciPy[5] and matplotlib[6] libraries were also made use of, NumPy for efficiency purposes and matplotlib and SciPy for plotting results.

2.0.1 Random/Senseless Agent

To implement the senseless agent, the environment was set with a problem ID in the range specified above, a stochastic environment and the reward for falling in the hole was set to 0.0. For each episode, and for each iteration within the episode, the agent will observe what its current state s is, the reward r for being in s and then its next action to move the agent into s' chosen at random from the possible actions available to the agent.

2.0.2 Simple Agent

The simple agent was given knowledge of the problem space before hand. This being the entire map of the environment and a heuristic $h(s)$ for each state s in the environment. $h(s)$ returns the straight line distance from s to the goal state. In order to maximize the agents performance measure, the choice of searching algorithm was obvious in this case - **A* search**[3]. Since the environment was not stochastic, and the agent was given knowledge of the environment beforehand. We know A* search to be complete and optimal[3] (if the heuristic is

admissible for tree-search and consistent for graph-search). This agent was run once in each problem ID because the environment is not stochastic and a deterministic algorithm is used, so the results should not change as all the same nodes are expanded in the same order in each episode.

2.0.3 Reinforcement Learning Agent

The choice of reinforcement learning algorithm used in this investigation was **Tabular Q-Learning**. Tabular Q-learning was chosen as the number of Q-values is not so large as to exhaust the state of the agent itself. The search space consists of only 64 states and in each state (apart from the terminal state) there is 4 actions to take (left, down, right, up). The choice for a Q-learning agent over other learning agents, is because a transition model of the environment does not have to be learned by the agent as the Q-value of being in a particular state s is directly related to s's neighbours.

The agent maintains a table of Q values Q, a state-action frequency table N, where in both cases the table is index by being in state s and taking action a. Other data that is provided to the agent is an exploration limit N_e, a learning rate α, a discount factor γ, a "bonus reward" R^+ before the agent has used up its exploration limit in s and taking a and the exploration function is defined as:

$$f(u, n) = \begin{cases} R^+ & \text{if } n < Ne \\ u & \text{otherwise} \end{cases}$$

Where the value that will be passed in as u is the $Q[s, a]$ value, n is the value $N[s, a]$. Upon each iteration in the episode, the agent will observe its current state s and the reward r for being in s. It decides its next action to take by selecting the action a' that has the highest $f(u, n)$ with the parameters listed above and in doing so, also updates the Q value for being in s and taking action a.

$$Q[s, a] \leftarrow Q[s, a] + \alpha(r + \gamma \ \text{argmax}_{a'} Q[s', a'] - Q[s, a])$$

3 Experiments and Results

Statistics collected from experiments:

- Maximum/Minimum mean reward from all episodes

- Mean reward collected per episode

- Covariance of the mean vector containing the mean rewards

- Goal loss

- Maximum/Minimum number of iterations to reach the goal

- Utility Loss

The reward for falling in a hole for the random and simple agent were both 0.0 and for the Q-learner it was -1.00. The random agent and the Q-learner were both run for 50000 episodes and allowed a maximum of 1000 iterations per episode. The extra parameters passed to the q learning agent are as follows:

$\alpha = 0.8$
$\gamma = 0.9$
$R^+ = 0.1$
$N_e = 1000$

3.0.1   Maximum/Minimum Mean Reward From All Episodes

Random Agent:

Problem ID	0	1	2	3	4	5	6	7
Max	0.001	0.001	0.001	0.003	0.001	0.001	0.001	0.005
Min	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00

Q Learning:

Problem ID	0	1	2	3	4	5	6	7
Max	0.00	0.063	0.09	0.08	0.07	0.08	0.063	0.11
Min	-0.14	-0.33	-0.25	-0.25	-0.33	-0.20	-0.33	-0.25

3.0.2   Mean Reward Per Episode

(Turn over for Graphs)

3.0.3   Covariance of the Mean Rewards

Random Agent:

Problem ID	0	1	2	3	4	5	6	7
Covariance	4.00×10^{-11}	9.99×10^{-11}	4.75×10^{-11}	6.20×10^{-10}	4.60×10^{-10}	1.01×10^{-09}	9.99×10^{-11}	9.61×10^{-9}

Q Learning:

Problem ID	0	1	2	3	4	5	6	7
Covariance	0.00012	0.0021	0.0023	0.0022	0.0058	0.0011	0.0021	0.0018

3.0.4   Goal Loss

Random Agent:

Problem ID	0	1	2	3	4	5	6	7
Num Goal	2	5	239	31	23	51	5	485

Q-Learning:

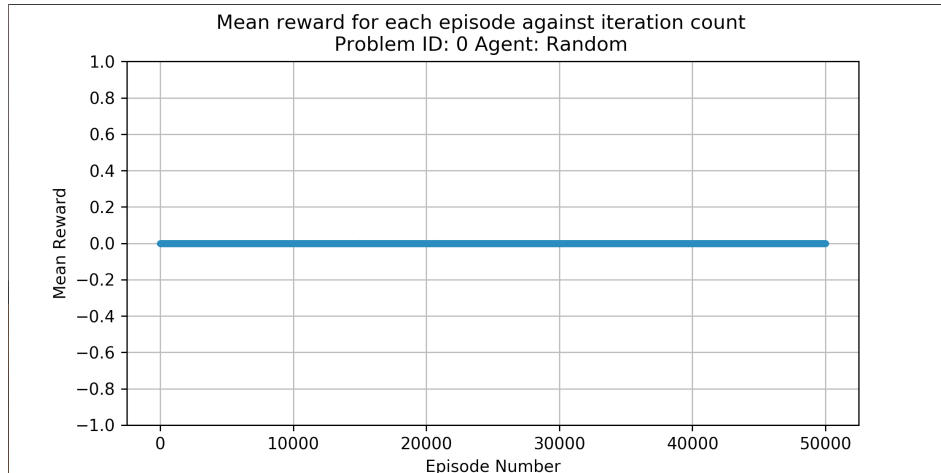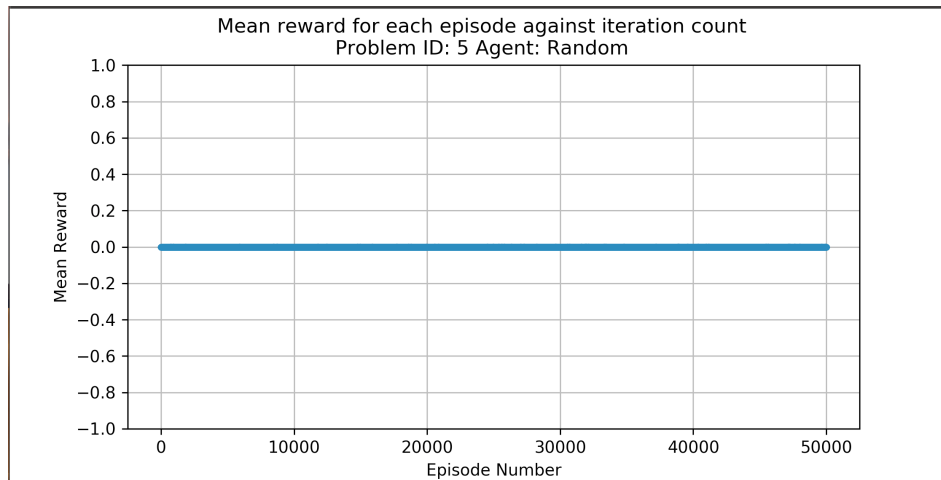| Problem ID	0	1	2	3	4	5	6	7
Num Goal	0	3676	14059	17413	13730	24154	1951	27157

Figure 1: Problem ID = 0, Random Agent
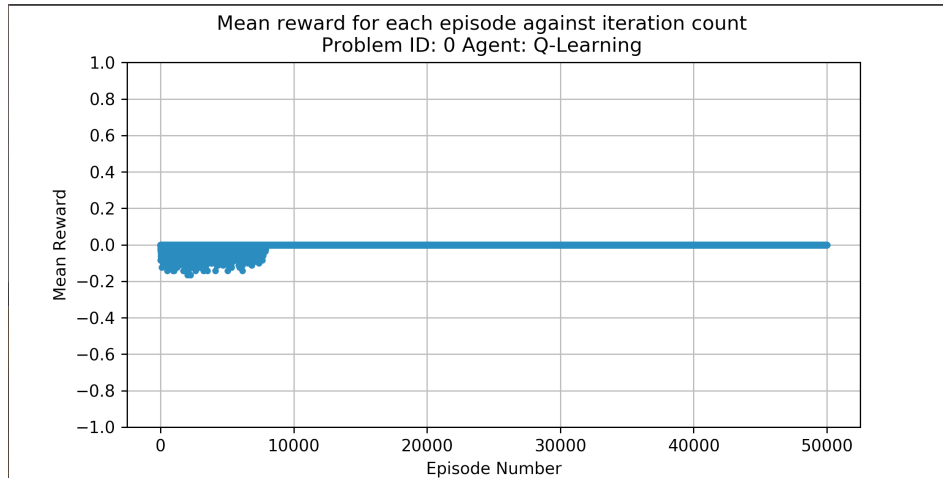

Figure 2: Problem ID = 3, Random Agent

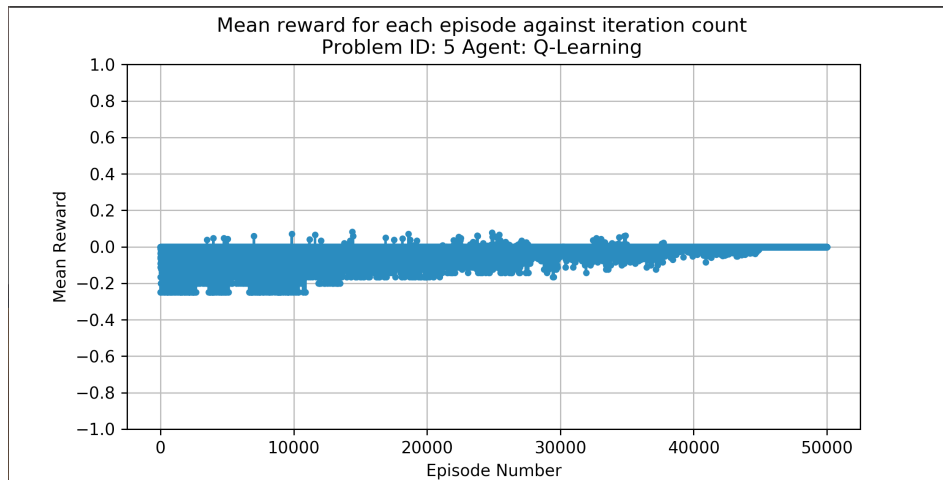Figure 3: Problem ID = 0, Q-learner

Figure 4: Problem ID = 5, Q-learner

3.0.5 Minimum/Maximum Number of Iterations to Reach the Goal

*if N/A appears it is because the agent did not reach that goal

Problem ID	0	1	2	3	4	5	6	7
Max iters	893	1000	1000	993	1000	1000	1000	1000
Min iters	454	232	489	789	1000	1000	1000	678

Simple Agent:

Problem ID	0	1	2	3	4	5	6	7
Iters	155	148	141	146	146	152	146	157

Q-Learning:

Problem ID	0	1	2	3	4	5	6	7
Max iters	N/A	1000	962	993	998	1000	994	655
Min iters	N/A	16	11	12	19	13	16	9

3.0.6 Utility Loss

The Value Iteration algorithm was used to compute the optimal policy for the underlying Markov Decision Process in the environment. These utilities were then compared with that of the Q-Learners (Turn over for results gathered):

4 Analysis of Results

4.0.1 Mean Rewards

As expected - the random agent's mean reward per episode stays mostly the same throughout all the episodes. This is expected as the agent is making non-deterministic decisions at each state it reaches and is not motivated by anything else such as the Q-learner, or the simple agent. Looking at the Q-learner, the agent clearly is learning what actions to take throughout the trials in order to maximize its performance measure. There is some variance in the mean rewards for the Q-learner, but this is to be expected since the environment itself is stochastic, therefore increasing the complexity of the environment as there is a transition model $P(s'|s, a)$ for each state in the environment.

4.0.2 Iterations

Looking at the tables, it is clearly noticeable that the simple agent will always find a solution to the goal. We know this to be optimal by the nature of A* search and the environment itself, is not stochastic. With regards to the random agent, the iterations to reach the goal cannot be really compared as the decision is just made at random and it is by chance, whether the agent reaches the goal or not. Over to the Q-learner, it is clear that the agent is learning throughout the episodes. The values for the maximum iterations to reach the goal is a

```
|||||||||||||||        State | Value itr | Q learning |  Diff

0:     +0.007    +0.000   +0.00696       32:    +0.006    +0.000   +0.00589
1:     +0.009    +0.000   +0.00930       33:    +0.007    +0.000   +0.00668
2:     +0.012    +0.000   +0.01246       34:    +0.007    +0.000   +0.00679
3:     +0.017    +0.000   +0.01654       35:    +0.000    +0.000   +0.00000
4:     +0.021    +0.000   +0.02051       36:    +0.224    +0.000   +0.22402
5:     +0.000    +0.000   +0.00000       37:    +0.335    +0.000   +0.33524
6:     +0.031    +0.000   +0.03120       38:    +0.158    +0.000   +0.15844
7:     +0.032    +0.000   +0.03231       39:    +0.103    +0.000   +0.10332

8:     +0.007    +0.000   +0.00694       40:    +0.004    +0.000   +0.00390
9:     +0.009    +0.000   +0.00925       41:    +0.000    +0.000   +0.00000
10:    +0.013    +0.000   +0.01254       42:    +0.000    +0.000   +0.00000
11:    +0.018    +0.000   +0.01812       43:    +0.107    +0.000   +0.10669
12:    +0.031    +0.000   +0.03133       44:    +0.320    +0.000   +0.31972
13:    +0.034    +0.000   +0.03429       45:    +0.735    +0.000   +0.73505
14:    +0.041    +0.000   +0.04051       46:    +0.000    +0.000   +0.00000
15:    +0.043    +0.000   +0.04312       47:    +0.054    +0.000   +0.05423

16:    +0.007    +0.000   +0.00691       48:    +0.003    +0.000   +0.00323
17:    +0.009    +0.000   +0.00903       49:    +0.000    +0.000   +0.00000
18:    +0.011    +0.000   +0.01125       50:    +0.013    +0.000   +0.01308
19:    +0.000    +0.000   +0.00000       51:    +0.036    +0.000   +0.03593
20:    +0.050    +0.000   +0.04967       52:    +0.000    +0.000   +0.00000
21:    +0.042    +0.000   +0.04248       53:    +1.795    +0.000   +1.79528
22:    +0.058    +0.000   +0.05766       54:    +0.000    +0.000   +0.00000
23:    +0.060    +0.000   +0.06013       55:    +0.023    +0.000   +0.02324

24:    +0.007    +0.000   +0.00707       56:    +0.004    +0.000   +0.00364
25:    +0.010    +0.000   +0.00961       57:    +0.005    +0.000   +0.00485
26:    +0.016    +0.000   +0.01595       58:    +0.008    +0.000   +0.00769
27:    +0.032    +0.000   +0.03232       59:    +0.000    +0.000   +0.00000
28:    +0.092    +0.000   +0.09179       60:    +9.999    +0.000   +9.99896
29:    +0.000    +0.000   +0.00000       61:    +5.249    +0.000   +5.24930
30:    +0.090    +0.000   +0.08962       62:    +2.250    +0.000   +2.24965
31:    +0.083    +0.000   +0.08267       63:    +0.000    +0.000   +0.00000
```

Figure 5: Utility Loss, Problem ID = 0, Q-learner

```
                                    State | Value itr |  Q learning |  Diff

0:      +0.000    +0.000    +0.00000         32:    +0.890    +0.000    +0.88998
1:      +0.144    -0.563    +0.70692         33:    +0.442    -1.015    +1.45677
2:      +0.131    -0.615    +0.74551         34:    +0.203    -1.290    +1.49345
3:      +0.098    -0.399    +0.49652         35:    +0.000    +0.000    +0.00000
4:      +0.070    -0.509    +0.57815         36:    +0.086    -0.478    +0.56490
5:      +0.000    +0.000    +0.00000         37:    +0.057    -0.935    +0.99236
6:      +0.016    -0.407    +0.42282         38:    +0.029    -0.654    +0.68301
7:      +0.014    -0.445    +0.45946         39:    +0.020    -0.481    +0.50104

8:      +0.239    -0.546    +0.78529         40:    +1.533    +0.000    +1.53257
9:      +0.205    -0.396    +0.60038         41:    +0.000    +0.000    +0.00000
10:     +0.162    -0.057    +0.21846         42:    +0.000    +0.000    +0.00000
11:     +0.097    -0.343    +0.44063         43:    +0.384    -1.290    +1.67353
12:     +0.065    -0.623    +0.68760         44:    +0.163    -0.881    +1.04396
13:     +0.036    -0.720    +0.75648         45:    +0.074    -0.732    +0.80609
14:     +0.023    -0.288    +0.31032         46:    +0.000    +0.000    +0.00000
15:     +0.017    -0.275    +0.29189         47:    +0.011    -1.027    +1.03724

16:     +0.354    -0.254    +0.60742         48:    +2.686    +0.000    +2.68630
17:     +0.282    -0.098    +0.37921         49:    +0.000    +0.000    +0.00000
18:     +0.204    -0.312    +0.51607         50:    +3.334    +0.860    +2.47424
19:     +0.000    +0.000    +0.00000         51:    +1.115    -1.290    +2.40531
20:     +0.050    -0.566    +0.61602         52:    +0.000    +0.000    +0.00000
21:     +0.033    -0.381    +0.41306         53:    +0.026    -1.290    +1.31631
22:     +0.023    -0.269    +0.29186         54:    +0.000    +0.000    +0.00000
23:     +0.017    -0.425    +0.44269         55:    +0.004    -1.290    +1.29450

24:     +0.544    +0.000    +0.54431         56:    +4.736    +0.000    +4.73580
25:     +0.380    -0.585    +0.96576         57:    +6.315    +0.000    +6.31474
26:     +0.236    -0.226    +0.46248         58:    +9.999    +0.000    +9.99896
27:     +0.091    -1.290    +1.38130         59:    +0.000    +0.000    +0.00000
28:     +0.068    -0.375    +0.44310         60:    +0.006    -1.290    +1.29591
29:     +0.000    +0.000    +0.00000         61:    +0.014    -1.179    +1.19286
30:     +0.021    -0.598    +0.61931         62:    +0.006    -1.078    +1.08358
31:     +0.018    -0.452    +0.46974         63:    +0.000    +0.000    +0.00000
```

Figure 6: Utility Loss, Problem ID = 6, Q-learner

lot higher compared to the minimum. Over time, the agent is learning better actions to take within each state in order to maximize its long term reward (expected utility).

4.0.3   Goal Loss

We know the simple agent to always reach the goal, so a comparison here with the other agents can't be really made, since the environment is deterministic in the simple agent case. However, looking at the random agent, the number of times the agent reaches the goal is very low in all cases, as previously mentioned, this is to be expected. From the different problem IDs, it varies quite largely depending on the difficulty of the problem. The Q-learner was able to reach the goal state in all problems, with signs of reaching a (most likely local) optimal policy when trying to reach a terminal state as indicated from the graphs and the agent is making it to the goal at a higher rate than the random.

4.0.4   Utility Loss

Looking at the tables, the agent never learns the true maximum expected utility in each state (and hence an optimal policy). The L_∞ norm of the difference being $+4.73580$. Looking at the graphs of the mean reward, we can see that the agent learns a policy which it may believe to be optimal and hence making the agent to make decisions that yes do give the agent a good average reward, but there is states the agent could also visit that may fully optimize its performance measure for the given environment, but is choosing not to. The number of episodes and iterations per episode is a vital factor within trying to work out the optimal policy and in this investigation may not have been high enough to allow the agent to explore the environment further and hence can be viewed as a limitation in this investigation.

5   Conclusion

In conclusion, it is clear that reinforcement learning can learn very close to optimal (if not optimal itself) policies to help the agent make rational decisions in each state it can end up in. Reinforcement learning is certainly the best approach out of the three agents that were used in environments that are not fully observable and are stochastic. However, if the environment is fully deterministic, simple agents are a very good option as you know the agent will succeed and there is a wide range of efficient algorithms that can be used in order for the agent to solve the problem.

References

[1] OpenAI Gym `https://gym.openai.com/`

[2] Artificial Intelligence: A Modern Approach Toolbox *Code implementations of the algorithms*

[3] Artificial Intelligence: A Modern Approach Pages 93-99 *A* search algorithm overview and proofs regarding it*

[4] NumPy Python Library `http://www.numpy.org/`

[5] SciPy Python Library `https://www.scipy.org/`

[6] matplotlib Python Library `https://matplotlib.org/`