AI Assessment: Learning in challenging environments with Q-Learning

Carlos Gemmell - 2209560

March 2019

1 Introduction

Algorithms that are capable of deciding optimally in unknown environments are useful in many situations. Navigation in uncertain terrain is one such example. The shortest path to an objective might not always be the optimal path since uncertain events can make different paths more risk prone to take. This means effective learning needs to take a more general approach to weighing risks and rewards.

In this work we attempt to solve a navigation problem based on the Open AI Frozen Lake environment. We use tabular Q-Learning and evaluate our results against random and ideal benchmarks.

2 Problem Definition

This problem consists of an 8x8 grid with holes spread out. We call a state in world a position on the grid with (X,Y) coordinates. An agent is initialized at a location in the grid. The goal state is another state in the grid. The objective for the agent is to reach the goal state. Reaching a state where a hole is located terminates the agent.

3 PEAS analysis

3.1 Performance

Given that the only reward in the environment is reaching the goal, we set success rate in reaching the goal as a measure for performance. There is no penalty for taking a step in the environment so the number of steps is not a factor, but could be an interesting contrasting measure. The number of steps to reach the goal will also be another measure to capture but will not influence model ranking.

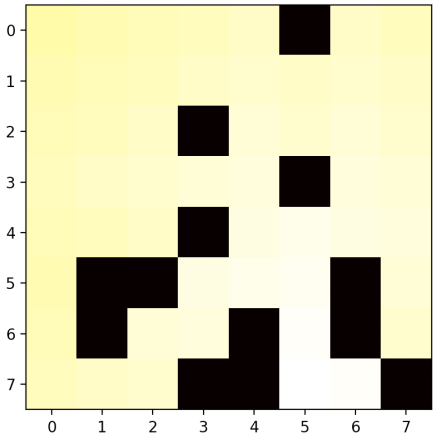


Figure 1: Map of the Loch Lomond environment.

3.2 Environment

The environment has 2 settings: stochastic and non-stochastic.

A non stochastic environment means that the agent will always perform the action it chooses. There will be no uncertainty with which state it will end up in based on what action it took.

The stochastic setting means there is uncertainty as to which state the agent will end up in after taking an action. Specifically, if the agent chooses to go in a particular direction, there is a 33% chance of the agent going where it said it would, and a 66% chance of going either side of it's chosen action. This makes decision making challenging. A reward of +1.0 for reaching the goal state is given. We set are able to modify the penalty for falling into a hole. We set this to values <0. Taking a step in the environment was no cost.

3.3 Actuators

An agent can perform 1 of a set of 4 actions: UP, DOWN, LEFT, RIGHT.

3.4 Sensors

For the stochastic version of the problem, there is no information available to the agent. For the other version, the entire map is visible (IE: hole and goal positions).

4 Agents

4.1 Random Agent

This agent serves as a baseline for our other results. It makes no informed decisions and takes an action randomly.

4.2 Simple Agent

This agent is placed in the non-stochastic environment and can see the entire map. For this reason we choose a searching algorithm capable of finding an optimal solution avoiding the holes. We choose A* [2], due to the understandability of the algorithm. We choose Manhatan distance for the distance heuristic.

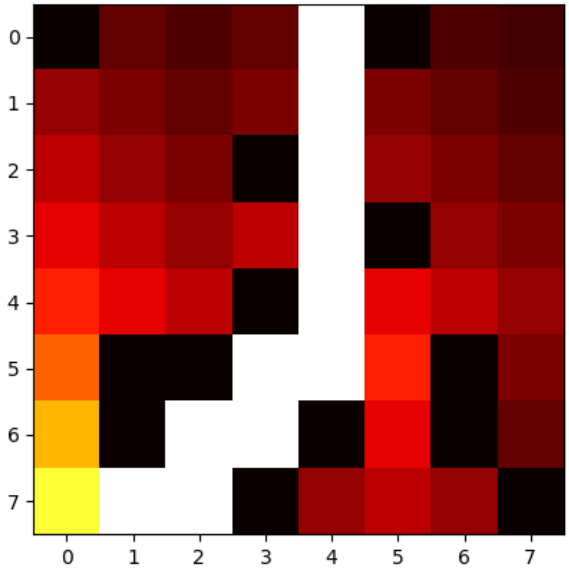


Figure 2: A-star path (White) overlaid over utility values obtained through value iteration (brighter is higher) on problem id=4.

4.3 Reinforcement Learning Agent

This agent is placed in the stochastic environment and has no information as to the placement of holes or the reward in the environment. We choose to have this agent use Tabular Q Learning [1]. This method consists in giving every possible action in a state a quality value. We store these values in a table with

every value initialized to 0. Actions are taken by choosing the action in a state with the maximum Q value. In order to effectively explore the environment and not exploit a single path. We include a random vector when choosing the action. This vector decays as the agent learns more, effectively reducing the exploration/exploitation traid off in favor of exploitation at the end of training.

```
action = np.argmax(Q[state,:] + np.random.randn(1,env.action_space.n)*(1./(iter+1)))
```

Figure 3: Decision step from Tabular Q-Learning algorithm.

Q values are updated when the agent takes a step and arrives in a new state. The maximum Q value in the new state is considered the "Value" of the state and is added with the reward received from entering the state. A discount rate an Learning rate are applied to first prefer sooner rewards over later one, then to control the speed at which the agent learns it's Q values.

```
Q[state,action] = Q[state,action] + eta*(reward + gma*np.max(Q[latest_state,:]) - Q[state,action])
```

Figure 4: Update step from Tabular Q-Learning algorithm.

Below we see the the chosen policy for our Q-learning agent. We can observe the arrows avoid the holes consistently yielding a 0% chance of falling in the hole immediately behind the arrow. In sections where the agent is between a wall and a hole, the agent has a clear prefrence to go in the direction of the wall rather than down. This can be seen as the agent atempting to minimise the risk of falling versus reaching the goal promptly

5 Experiments

The random agent and simple search algorithm have no experiments performed in this work.

We choose to modify the penalty for entering a hole for the Q-learning algorithm. This is of interest to us since the traid-off between being cautious around holes and reaching the goal fast. We choose a range of values for the penalty ranging from -0.01 being insignificant against the goal to -100 having the opposite impact.

5.1 Experimental Setup

Every algorithm is run for 5k episodes with each episode consisting of a maximum of 500 iterations.

Learaning rate is 0.6, and discount factor is 0.9.

Running these setting for all environments can take up to 10 minutes, I multi-thread the problems to get almost a 10X speedup. Below is the command to run the full evaluation. This generates graphs for all problem ids.

```
$ python3 run_eval.py 5000
```

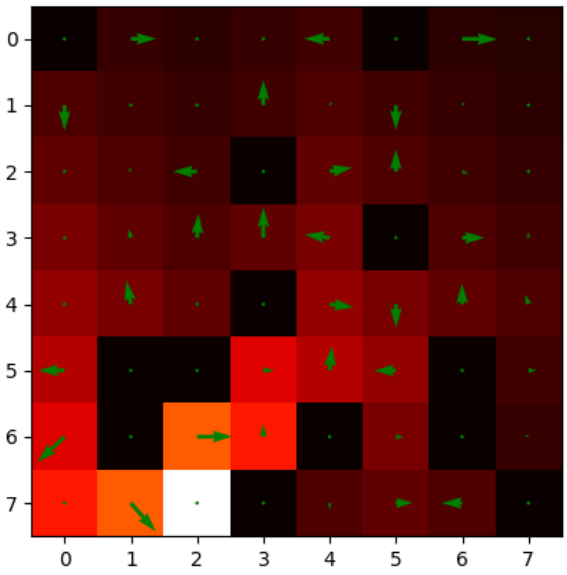


Figure 5: Q-learning policy (Arrows) overlaid over utility values obtained through value iteration (brighter is higher) on problem id=6.

6 Results

Here are the results for our penalty search on the Q-Learning Agent.

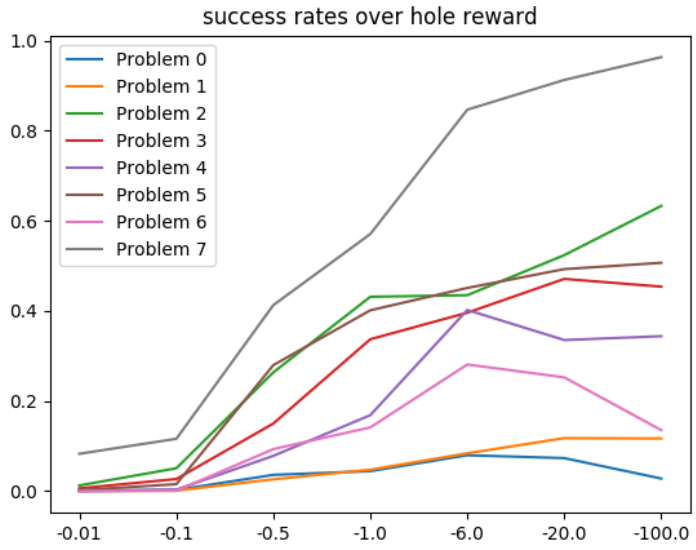
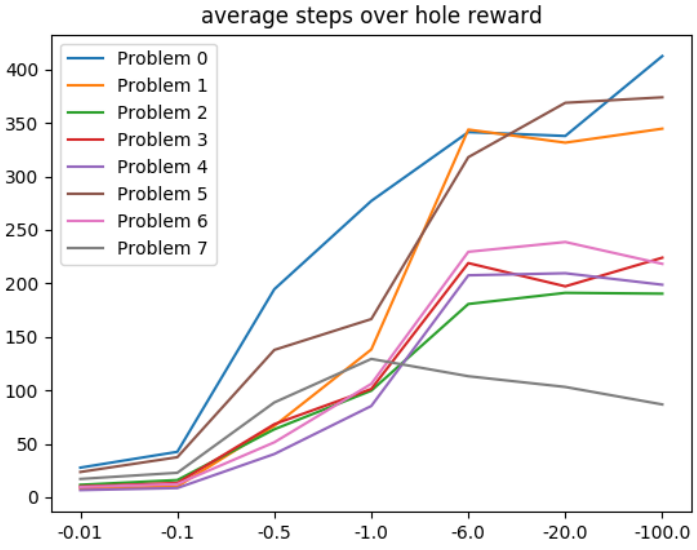
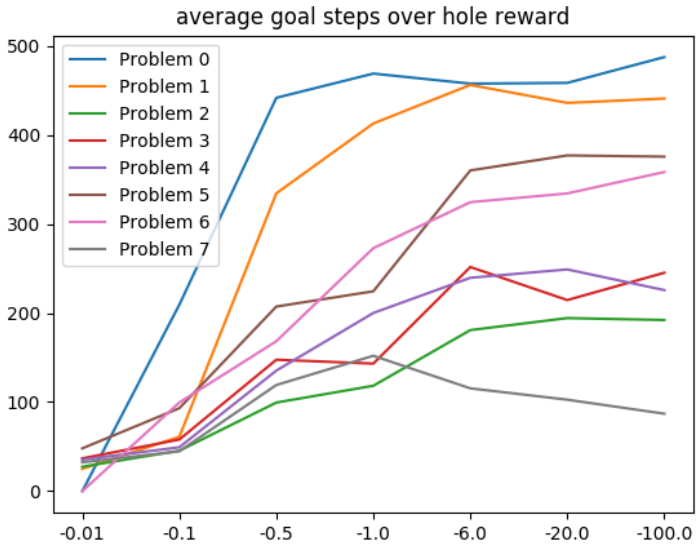

success rates over hole reward
Problem 0
Problem 1
Problem 2
Problem 3
Problem 4
Problem 5
Problem 6
Problem 7
1.0
0.8
0.6
0.4
0.2
0.0
-0.01
-0.1
-0.5
-1.0
-6.0
-20.0
-100.0

Figure 6:


average steps over hole reward
Problem 0
Problem 1
Problem 2
Problem 3
Problem 4
Problem 5
Problem 6
Problem 7
400
350
300
250
200
150
100
50
0
-0.01
-0.1
-0.5
-1.0
-6.0
-20.0
-100.0



We see clear increase in success when the penalty increases. The amount of steps increases as the penalty increases. We can think of this as the agent trying to avoid the holes at all cost and eventually reaching the goal through its avoiding strategy.

Taking this process into account, we take the best value for the penalty by training the algorithm for a specific amount of time, then once it has converged, we capture the mean success rate for the rest of the iterations. We usually find the algorithm has converged at 200 episodes.

Figure 7: Comparison of all 3 Agents with a hole reward of -100. Vertical bar indicating metric selection threshold.

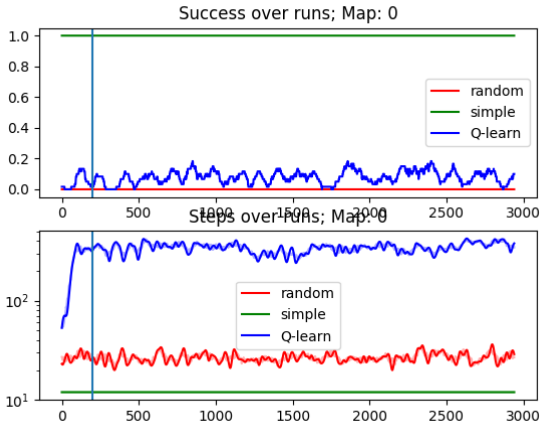


Figure 8: Comparison of all 3 Agents with a hole reward of -6. Vertical bar indicating metric selection threshold.

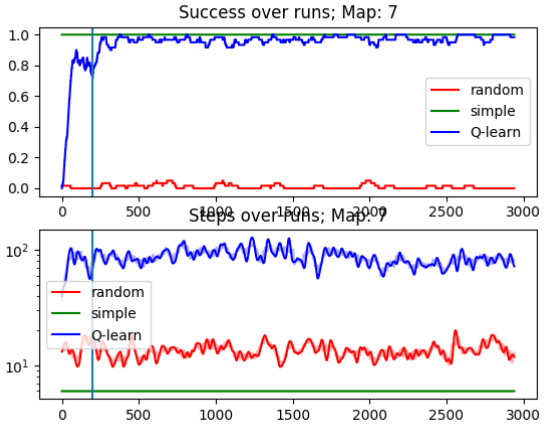


Figure 9: Comparison of all 3 Agents with a hole reward of -100. Vertical bar indicating metric selection threshold.

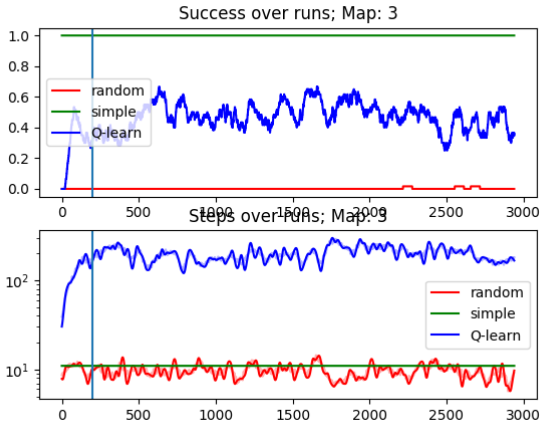


Figure 10: Comparison of all 3 Agents with a hole reward of -20. Vertical bar indicating metric selection threshold.

7 Conclusion

We can see that optimal path to the goad obtained by the A Star algorithm is much shorter that that obtained by the Q-Learning algorithm. However, the deterministic environment is the only reason the algorithm is able to outperform. Given the uncertainty in the stochastic environment, the Q-Learning algorithm

manages to perform best by avoiding the holes as much as possible and only arriving at the goal after a large amount of steps. However this is a valid strategy given the problem definition since the amount of steps taken does not incur any penalty.

We see the random agent performing poorly every problem with it only making a few successful runs in problem 7.

References

[1] Marco Dorigo and Hugues Bersini. A comparison of q-learning and classifier systems. *From animals to animats*, 3:248–255, 1994.

[2] František Duchoň, Andrej Babinec, Martin Kajan, Peter Beňo, Martin Florek, Tomáš Fico, and Ladislav Jurišica. Path planning with modified a star algorithm for a mobile robot. *Procedia Engineering*, 96:59–69, 2014.