Reinforcement Learning agent in Wumpus World

Abhee Parekh, Rishabh Gupta, Saumil Dixit, Sukhpreet Anand

Abstract - The aim of this project is to integrate reinforcement learning to manage our agent in the wumpus world environment. In precise terms, we are implementing a q-learning agent to play the game of wumpus world with a noisy action model, thereby turning our logical agent into a learning one. Moreover, the goal of our agent is to make decisions based on a policy that permits it to grab gold while evading the wumpus and the pits present in the environment. The agent learns its moves through repetitive training which takes into account the characteristics of the world. In this report we will compare the performance of our reinforcement learning agent under a noisy action model with that of the hybrid logic agent. Furthermore, we will provide detailed statistical analysis for the results along with an analysis of policy convergence for our agent.

I. INTRODUCTION

Reinforcement learning can be implemented to control an agent by allowing it to interact with the environment and gradually learn from the inputs that it receives. Using an active reinforcement learning approach, our agent works towards learning an optimal policy which enables it to collect the best possible reward over the passage of time. Q-learning is a popular technique for carrying out reinforcement learning. It is an off-policy technique which essentially means that it works without the need for a model which makes it different from other techniques which make use of the bellman equation, such as value iteration and policy iteration. The task of a q-learning agent in this scenario is to find an optimal policy for a given layout which maximizes its reward. This means to explore the grid for the gold, grab the gold and then climb out of the cave (grid) in the shortest time, without encountering the wumpus or a pit.

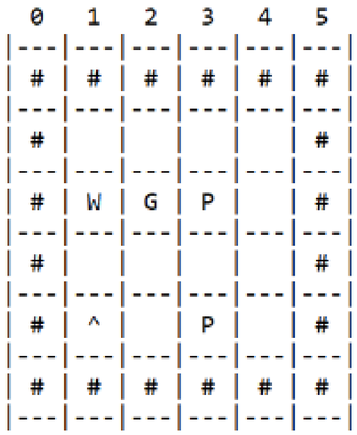
The environment that the agent will be operating in is a version of the Wumpus world game in which it tries to explore the cave, which is represented as a NxN grid. The characteristics of the world are : partially observable, stochastic, sequential, static, discrete and single agent. The elements of this world consist of a wumpus, gold and bottomless pits. As opposed to the deterministic version of the original Wumpus world game, we have adopted a stochastic version of this environment where the outcomes of actions will not be completely deterministic. For instance, if the agent takes an action to go forward, then due to the stochastic nature of the environment, there is a possibility that instead of actually going forward, it might end up in either the right or the left cell adjacent to the cell in which the agent is in. The focus of our agent will be to maximize its reward by following the optimal policy. The actions available to the agent to explore the environment are 'TurnRight' and 'TurnLeft' to change the direction of the agent, 'Forward' to move ahead in the direction the agent is facing, 'Grab' to grab the gold, 'Shoot' to shoot the wumpus with the arrow, 'Climb' to come out of the grid. The constraint is that the agent has only one arrow to shoot the Wumpus and the agent can climb out of the grid only from the initial location. For each time step, the agent is inside the grid, it will receive a negative reward of -1. If the agent uses an arrow it will immediately receive an additional reward of -10. If the agent dies by going into Wumpus or falls into the pit it will receive a reward of -1000, thereby ending the game. If the agent gets the gold and successfully comes out of the grid then it will receive a reward of +1000. Also, according to the game specification of Wumpus World, there can be only one Wumpus. The agent tries to form a policy that will maximize the rewards. Agent receives information about the world through the given percepts :

<Breeze, Stench, Scream, Glitter, Bump>

which it can utilize to perceive different characteristics of the environment. To elaborate, if the agent is adjacent to the wumpus, it will receive a stench. If the agent is at the same location as the gold, it will receive glitter. And if the agent is adjacent to a bottomless pit, it will receive a breeze. When the agent shoots down the Wumpus, it lets out a scream which can be heard across the grid. Moreover, a bump is perceived when the agent hits a wall while attempting to go forward. We propose to integrate the q-learning method to control our agent through the wumpus world so that it can learn to reason about the environment. The goal here is to show that through multiple trainings, the learning agent should be able to maneuver through complicated situations where an agent, working purely on formal logic, might not be able to succeed. And Q learning agent converges to an optimal policy, even if the agent acts sub-optimally in the training.

II. TECHNICAL APPROACH

We have implemented a q-learning agent in the wumpus world environment by extending the "Hunt the Wumpus AI Project" by Clay Morrison (University of Arizona) [4] and "Reinforcement Learning Agent" related files from UC Berkeley AI Project [3]. Our agent aims to navigate through the 4x4 grid version of the environment by moving into an adjacent square in any of the four directions. To implement q-learning for this task, we need a description of states, actions and rewards associated with those actions.



```
  0   1   2   3   4   5
|---|---|---|---|---|---|
| # | # | # | # | # | # | 5
|---|---|---|---|---|---|
| # |   |   |   |   | # | 4
|---|---|---|---|---|---|
| # | W | G | P |   | # | 3
|---|---|---|---|---|---|
| # |   |   |   |   | # | 2
|---|---|---|---|---|---|
| # | ^ |   | P |   | # | 1
|---|---|---|---|---|---|
| # | # | # | # | # | # | 0
|---|---|---|---|---|---|
```

Figure 1 : A 4x4 layout of the wumpus world environment showcasing the locations of our agent at (1, 1), wumpus at (1, 3), gold at (2, 3), and pits at (3, 1) and (3, 3)

The state for our agent can be represented as shown below :-

<x, y, orientation, has_gold, wumpus_alive>

where *(x, y)* represents the current location of our agent, *orientation* represents the direction in which the agent is facing, which could be further delineated as : 0 for North, 1 for West, 2 for South, and 3 for East. Furthermore, *has_gold* represents a flag that would be useful in determining whether the agent is carrying the gold or not. The flag is set to 1 when the agent is carrying the gold and is set to 0 when the agent is not carrying the gold. Similarly, *wumpus_alive* represents a flag which helps in determining whether the wumpus is alive in the environment or not. Initially, it is set to 1, and will be set to 0 after our agent shoots down the wumpus. As previously mentioned, the *Scream* percept is used for determining if the wumpus has been killed which consequently sets the flag 0.

The agent is capable of choosing from the following set of six legal actions :-

<TurnRight, TurnLeft, Forward, Grab, Climb, Shoot>

where the actions include moving or shooting in any of the four directions. If the agent decides to shoot in the direction of the wumpus, the wumpus will die making it safe for the agent to enter the cell which the wumpus inhabited.
The parameters which control the behavior of the agent are :-
Exploration factor (ε) - It determines the probability of random action the agent chooses at any given step.
Learning Rate (α) - It determines the weight of the new sample in the existing policy.
Discount Factor (γ) - It determines how much the agent cares about the rewards in the distant future as compared to immediate future.

Owing to the stochastic nature of the environment, the *Forward* action achieves its intended effect with some probability, say p. For the rest of the time, the same action moves the agent at right angles to the intended direction of motion with probability 1-p. Moreover, stochasticity could be controlled by changing the action probabilities. More conclusively, higher the probability associated with action *Forward*, lower is the stochasticity involved while executing that action. The Q-learning formula can be given as :-

$$Q(s,a) \leftarrow (1-\alpha)*Q(s,a) + \alpha*[R(s,a,s') + \gamma*\max_{a'} Q(s',a')]$$

where *s* is the previous state and *s'* is the newly generated successor state after an action *a* is performed. Furthermore, *R(s,a,s')* is the reward collected, and *α* and *γ* are the learning parameters. To elaborate, *α* is the learning rate and *γ* is the discount factor.

The q-learning algorithm [1][2] that we have administered works as follows :-

- We first train the agent. For this we run minimum trials without convergence, then let it converge. It will stop training, once it has trained for max trials.
- Each trial is an episode which is capped at 1000 steps
- In the current state, the agent chooses the optimal action (based on Q(s,a) values calculated so far) or a random action based on epsilon value (higher epsilon means more random actions) to perform.
- After choosing this action, the agent will immediately get the current performance measure. Based on current and previous performance measures, the reward is calculated. Based on reward, state and action, our agent updates the Q(s,a) values.
- The convergence of the policy is checked after the minimum number of trials has been reached. This assumption has been kept to assure that there is no premature convergence due to which we might obtain a sub-optimal policy.
- After the minimum number of trials have been performed, we check for convergence after each training episode. If we get two consecutive policies which are the same and the q-values associated with both the policies don't change significantly (indicated by the delta value), we assume that the policy convergence has been achieved and the training will stop.
- Then after generating policy we test the policy by running it many times and then averaging the scores. If the steps in a test are going beyond a 1000 steps, we ignore that test. If too many tests are overstepping beyond 1000 steps, we conclude that the policy is not optimal. But if only some tests are going beyond 1000, then we say it is because of stochasticity.

III. RESULTS

The agent was allowed to carry out exploration in the grid for a fixed number of iterations. Its performance was was evaluated and observations were made as shown below :-
Default Parameters : α = 0.2, ε = 0.05, Stochasticity (0.1, 0.8, 0.1), γ = 0.8

1. Effect of Discount Factor (Gamma) on Q-Learning Agent

Gamma (γ)	Runs to reach Convergence	Average Final Score
1.0	544	974
0.9	6102	954
0.8	4025	974
0.7	4007	974
0.6	8168	919

Figure 2 : Runs required to achieve convergence and average final score for different values of γ

As we can see from above table, for gamma = 1, it needs only 500 runs to create optimal policy, this is because it is not discounting the reward of getting gold which it receives only at the end of the episode which can be many steps beyond. If we decrease gamma, the value of getting gold and getting out decreases drastically because for each step, its value is discounted by a factor of gamma.

2. Effect of Learning Rate (Alpha) on Q-Learning Agent

Alpha (α)	Runs to reach Convergence	Average Final Score
0.1	8051	974
0.2	4035	974
0.3	4151	974
0.8	2026	974
1.0	Inconclusive	Inconclusive

Figure 3 : Runs required to achieve convergence and average final score for different values of α

As we can see from the above table, as the value of alpha increases the number of trials required to get optimum policy will decrease. For alpha = 1, the past policy will be overwritten by the newer policy. This will result in no conclusion.

3. Effect of Exploration Factor (Epsilon) on Q-Learning Agent

Epsilon (ε)	Runs to reach Convergence	Average Final Score
0.0025	8706	974
0.025	4834	974
0.05	4609	974
0.1	4543	973
0.5	4502	974

Figure 4 : Runs required to achieve convergence and average final score for different values of ε

As epsilon increases, from a really low value to a comparatively higher value, the number of trials needed reduces drastically. After a certain value i.e. ε = 0.025, there is no significant difference in the number of trials though.

4. Effect of Stochasticity on Q-Learning Agent

Stochasticity (s)	Runs to reach Convergence	Average Final Score
(0.2, 0.6, 0.2)	8007	971
(0.1, 0.8, 0.1)	4011	974
(0.05,0.9,0.05)	5598	975
(0.0, 1.0, 0.0)	223	987

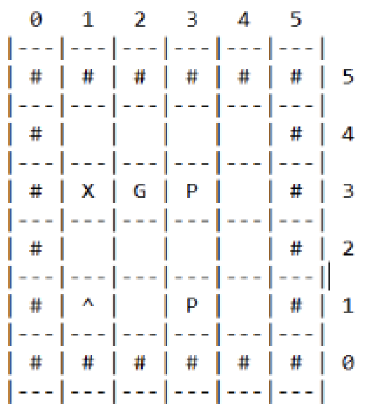
Figure 5 : Runs required to achieve convergence and average final score for different values of α

As the stochasticity of the environment is reduced, the number of trials required to generate an optimum policy is reduced. Results are not consistent when we are varying all the parameters due to uncertainty in the environment. Even if a good policy is reached it is still possible for the agent to die or to go beyond 1000 steps. This is due to stochastic environment.

Performance of Q-Learning Agent on Layout with varying parameters :-

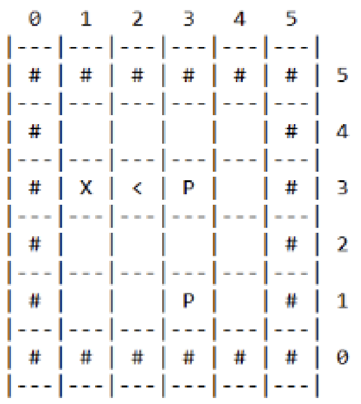
For the layout given in **Figure 1**, the agent undergoes different behavior for different parameters provided to it. (Note: The default parameters for α is 0.2 and γ is 0.8 for the below observations) :-

Observation 1 : High stochasticity, high exploration: for s = [0.1,0.8,0.1] and **ε = 0.4**,, due to high stochasticity, the Agent uses the arrow to kill the Wumpus in the first time step.



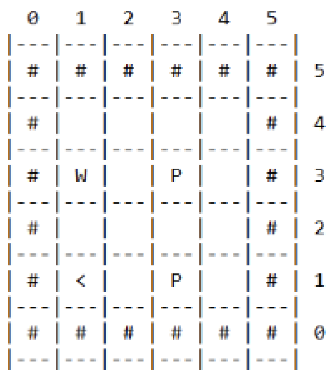
```
  0   1   2   3   4   5
|---|---|---|---|---|---|
| # | # | # | # | # | # | 5
|---|---|---|---|---|---|
| # |   |   |   |   | # | 4
|---|---|---|---|---|---|
| # | X | G | P |   | # | 3
|---|---|---|---|---|---|
| # |   |   |   |   | # | 2
|---|---|---|---|---|---|
| # | ^ |   | P |   | # | 1
|---|---|---|---|---|---|
| # | # | # | # | # | # | 0
|---|---|---|---|---|---|
```

This is because after the agent grabs the gold, at position (2,3), if the Wumpus was alive, a forward action from (2,3) to (2,2) would have had a 0.2 probability of the Agent being killed by the Wumpus or falling into the pit. However, on killing the Wumpus, the agent simply takes a forward action from (2,3) to (1,3) to avoid falling into the pit in such a situation. Due to high exploration, it was also able to explore the optimal policy quickly, which has 16 time steps.



```
  0   1   2   3   4   5
|---|---|---|---|---|---|
| # | # | # | # | # | # | 5
|---|---|---|---|---|---|
| # |   |   |   |   | # | 4
|---|---|---|---|---|---|
| # | X | < | P |   | # | 3
|---|---|---|---|---|---|
| # |   |   |   |   | # | 2
|---|---|---|---|---|---|
| # |   |   | P |   | # | 1
|---|---|---|---|---|---|
| # | # | # | # | # | # | 0
|---|---|---|---|---|---|
```

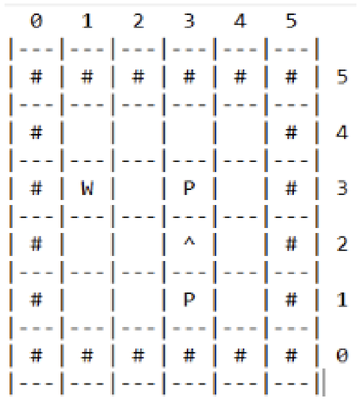
Observation 2 : Low stochasticity, high exploration:
For s = [0.0,1.0,0.0] and **ε=0.4**, due to low stochasticity, the agent doesn't waste time in killing the Wumpus. Because in this case on undergoing a forward action from location (2,3) to (2,2), it has 0 probability of being killed by a Wumpus or falling into a pit. Thus, it follows a straightforward path to fetch the gold and comes back out of the cave. This can be show by the final state of the environment after the agent climbs out of the cave :-



```
  0   1   2   3   4   5
|---|---|---|---|---|---|
| # | # | # | # | # | # | 5
|---|---|---|---|---|---|
| # |   |   |   |   | # | 4
|---|---|---|---|---|---|
| # | W |   | P |   | # | 3
|---|---|---|---|---|---|
| # |   |   |   |   | # | 2
|---|---|---|---|---|---|
| # | < |   | P |   | # | 1
|---|---|---|---|---|---|
| # | # | # | # | # | # | 0
|---|---|---|---|---|---|
```

Due to high exploration, the optimal policy takes 13 steps which is faster than in observation 1 (due to low stochasticity).

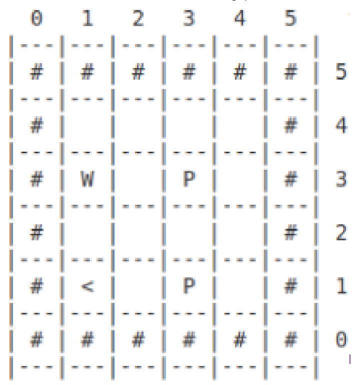
Observation 3 : High stochasticity, low exploration:
For s = [0.1,0.8,0.1] and **ε=0.05**, due to very low exploration value, when the agent goes to state (3,2) between the two Pits, it stays there indefinitely and continuously undertakes the 'Climb' action for over 1000 time steps as can be shown below :-



```
  0   1   2   3   4   5
|---|---|---|---|---|---|
| # | # | # | # | # | # | 5
|---|---|---|---|---|---|
| # |   |   |   |   | # | 4
|---|---|---|---|---|---|
| # | W |   | P |   | # | 3
|---|---|---|---|---|---|
| # |   |   | ^ |   | # | 2
|---|---|---|---|---|---|
| # |   |   | P |   | # | 1
|---|---|---|---|---|---|
| # | # | # | # | # | # | 0
|---|---|---|---|---|---|
```

This happens because due to high stochasticity the agent does not attempt to risk going to location (2,2) and (4,2) on forward action from location (3,2) action because it will have a 0.2 probability of falling in either of the Pits. Also, due to low exploration in this situation, it will further not try to explore other actions at that position because continuously trying to 'Climb' would keep the Agent alive longer and prevent it from getting a -1000 reward by falling into a pit. Hence high stochasticity and low exploration can make our agent get stuck in certain pitfalls.

Observation 4 : Low stochasticity, low exploration:
For s = [0.0,1.0,0.0] and **ε=0.05**, due to low exploration, but no stochasticity, the optimal policy is found in only 200 trials, the optimum policy has 13 steps which is faster than in observation 1 (due to low stochasticity).



```
  0   1   2   3   4   5
|---|---|---|---|---|---|
| # | # | # | # | # | # | 5
|---|---|---|---|---|---|
| # |   |   |   |   | # | 4
|---|---|---|---|---|---|
| # | W |   | P |   | # | 3
|---|---|---|---|---|---|
| # |   |   |   |   | # | 2
|---|---|---|---|---|---|
| # | < |   | P |   | # | 1
|---|---|---|---|---|---|
| # | # | # | # | # | # | 0
|---|---|---|---|---|---|
```

Even if the exploration is low, the low stochasticity overpowers the low exploration value and would make us find the optimal policy quicker.

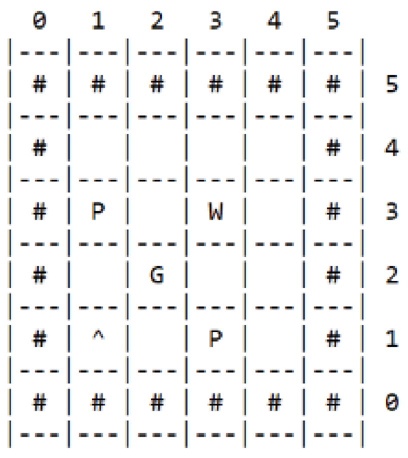
Observation Summary:

High stochasticity High exploration Sometimes desirable	Low stochasticity High exploration Highly desirable
High stochasticity Low exploration Strictly undesirable	Low stochasticity Low exploration Desirable

Comparison of Q-Learning Agent with Knowledge-Based Agent :-

The knowledge based agent tries to find possible safe locations from its knowledge base. If there are no safe locations to be moved to from the current location, then the knowledge based agent will try to shoot the arrow. If the agent hears scream, then it will update its knowledge base with wumpus alive as false. If no safe locations are updated in the knowledge base, then the agent takes a risk and goes to an unsafe location.

For the original layout the hybrid agent always gets a 983 score, but it takes a lot of time, because as the steps increase the clauses in its knowledge base increase exponentially. The Q-learning agent gets an average score of 974 in a stochastic environment after training and generating a good policy, fairly quickly as compared to hybrid agent. Consider a scenario as shown in figure 2. The knowledge based agent will detect stench at (1,2) and (2,1) and will not be able to determine any safe locations. Thus it will take risk from location (2,1) by moving forward and end up falling in a pit at (3,1) always. So it gets a score of -1019 for this layout. But the q learning agent after getting sufficient amount of training will form a policy to get the gold and climb out of the grid. The average score of q learning agent after 4000 training, in a stochastic environment of 0.8 is 749.



```
  0   1   2   3   4   5
|---|---|---|---|---|---|
| # | # | # | # | # | # | 5
|---|---|---|---|---|---|
| # |   |   |   |   | # | 4
|---|---|---|---|---|---|
| # | P |   | W |   | # | 3
|---|---|---|---|---|---|
| # |   | G |   |   | # | 2
|---|---|---|---|---|---|
| # | ^ |   | P |   | # | 1
|---|---|---|---|---|---|
| # | # | # | # | # | # | 0
|---|---|---|---|---|---|
```

Figure 6 : Layout used for comparing the performance of the q-learning agent with that of the hybrid logic agent. The agent is located at (1, 1), wumpus at (3, 3), gold at (2, 3), and pits at (1, 3) and (3, 1).

IV. CONCLUSIONS

In our report we presented a q-learning based reinforcement learning approach in the wumpus world with a noisy action model. It can be observed from the training that our learning agent is able to find gold while successfully evading the wumpus and pits. It generates different optimum policies based on different values of Gamma and Stochasticity. Number of trials to generate an optimum policy depends on various factors such as gamma, alpha, epsilon and stochasticity. We also showed in the results a scenario where the learning agent was successful in an environment where the knowledge based logic wasn't able to succeed.

V. TEAM EFFECTIVENESS

SAUMIL DIXIT : Worked on preparing the final project report. Performed training experiments and statistical analysis on the working of the q-learning agent for different parameters and layouts. Compared the performance of the q-learning agent with the hybrid logic agent under the noisy action model.

SUKHPREET ANAND : Worked on integrating hybrid logic agent from 'The Hunt the Wumpus' project by University of Arizona with our q-Learning agent project, including integration testing. Worked on statistical analysis of data for different layouts and with different parameters. Created multiple layouts for assessing performance of the q-learning agent

ABHEE PAREKH : Implemented code for updating q-values, fetching q-values based on bellman equations and getting set of legal actions defined by the game. Conducted training experiments for statistical analysis of performance of q-learning agent for different parameters and layouts.

RISHABH GUPTA : Worked on implementing the agent program function which returns the required action based on the q model prepared so far. Modified the parameters to be considered in a single state. Prepared code for conducting trainings along with convergence and executing the final run. Conducted statistical analysis for different stochastic values and analysis of performance between hybrid logic agent and q-learning agent.

REFERENCES

[1] Stuart Russell and Peter Norvig, "Reinforcement Learning" in Artificial Intelligence : A Modern Approach, 3 r d ed., NJ, USA: Prentice Hall Press, 2015, pp. 846-867.

[2] Richard S. Sutton and Andrew G. Barto, "Temporal-Difference Learning" in Reinforcement Learning: An Introduction, 2nd ed.,London, England: The MIT Press 2015, pp. 143-167.

[3] "Reinforcement Learning" in Berkeley AI Materials, http://ai.berkeley.edu/reinforcement.html

[4] "Using Logic To Hunt The Wumpus" by Clay Morrison http://www.sista.arizona.edu/~clayton/courses/ai/projects/wumpus/