Reinforcement Learning Agent in Wumpus

Dhaivat Dholakiya, Karan Shah, Siddharth Joshi, Venkatesh Vaidyanathan

Abstract—

In this project we aim to compare the results and convergence rates of Propositional Logic and Reinforcement Learning in a known environment infamously known as Wumpus World. Propositional Logic has served as the bedrock of AI in its infancy, giving it a certain level of autonomy at its birth. But as time goes on we have been able to build better systems and methods and the latest kid on the block is Reinforcement Learning. In the following chapters we aim to explore in details the working, nuances and shortcoming of both Propositional Logic and Reinforcement Learning.

In a game setting we are introduced to logical agents using Wumpus as a medium of understanding. We aim to make our agent reason in its world rather than using a predefined KB. We aim to create an agent which can reason its way to its objective in a unknown environment (for the agent) without the use of propositional logic.

I. Introduction

Propositional Logic is a form of logic which is based on a truth value basis or boolean logic. It is also one of the most highly used forms of logic. It involves the combination of propositions as its basic building blocks which lead to some powerful logical chains. The most important application of propositional logic lies in the field of Artificial Intelligence in the domains of planning, problem solving and primarily used for decision making. Going beyond the truth aspect propositional logic also helps a great deal in certainty and uncertainty, and has been the foundation of machine learning.

There are other kinds of propositional logic which the hybrid wumpus agent relies on, such as predicate logic or first-order logic. The primary premise of propositional logic is logical connectives. Logical connectives are what help us connect two or more propositions together to obtain a logically cohesive sentence. A few of the logical connectives used in the construction of the hybrid agent are negation, conjunction, disjunction, implication and biconditional.

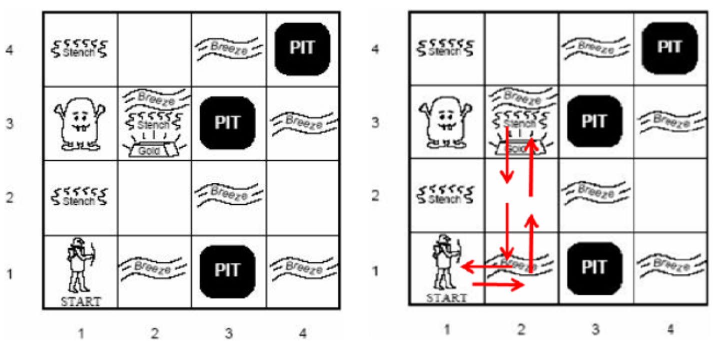
Although everything seems good so far, there are some fallacies and shortcomings when it comes to propositional logic. The first fallacy being that not all relations can be expressed in the form of propositional logic. Second proportional logic is very limited in terms of its expressive and encapsulating power. The final one being that we cannot describe statements in terms of their properties or logical relationships.

To overcome these shortcomings and go above and beyond we employ a relatively novel learning method known as reinforcement learning. Reinforcement Learning involves the use of training models to make a sequence of decisions. Out agent tries to successfully reach a goal in an uncertain and to an extent complex environment. The agent employs a trial and error method. At first losing the game in a few straight rows and gradually learning the games nuances and in the end becoming really good at the game.

The desired behavior is obtained by setting a reward policy and allows the agent to agent to learn by itself from the word go. It is unto the agent to determine how to maximize the reward function for the given task at hand. By employing the power of a few RL algorithm and a given set number of trials the agent is able to achieve its proficiency.

II. Technical Approach

The specific approach we have take up in reinforcement learning is Q-Learning. The reason for adoption in specific to this method is the face that Q-Learning does not require a model like other reinforcement learning techniques and it is reliant or dependent upon the implicit use of bellman equations and the nuances it brings along with it such as value and policy iteration. The primary goal of the Q-Learning agent we have devised is to find the gold and exit the maze without encountering a wumpus or a pit. This is the same approach which is imbibed in the hybrid agent which required the building of a KB with propositional logic.



Now if we were to look at the environment it is a 2-D grid world with 16 cells. Each cell may either be empty, contain a pit or wumpus which is dangerous or contain the gold which is what the agent is striving for. The agent may traverse through the environment by moving through either the up, down, left or right. If the agent enters a cell which contains a pit or the wumpus it will lead to the termination of the agent and the game ends with a loss. If the agent does enter a cell which is adjacent to either a pit or wumpus, the agent will be able to perceive a breeze or a stench respectively.

The task of the agent is to try its best too safely traverse to the gold by using these percepts to keep safe. To employ Q-Learning on this take we use the following tuple:
$(X, Y, arrow, scream, locationhistory)$
The X,Y give us the current location of the agent in the map. The arrow gives us a percept of whether the arrow is still present or if it has been used. The scream percept is present to give us a sense of whether the wumpus has been killed or not. The location history is the stored percept values of what the agent is aware of each visited state.

The Q-Learning approach that we have taken is executed as follows:
- Select the best action from a given sequence of actions for the current state.
- Execute the best known action for the given situation and receive a reward or penalty for that action.
- Once the reward has been executed we enter a new state or the terminal state.
- In between the change of states we update the policy and reassign new rewards for new states.

The policy which yields the optimal reward is updated in the Q-table and constantly executed. A Q-table is nothing but a collection of the state action pairs with certain q-values assigned to each pair, which gives the optimal nature of the pair. At the start of the iterations the table is fairly uniform in its state-action pairs but as iterations execute the optimum policy is eventually found using the following formula:
$Q(S_t, a) \leftarrow (1-\alpha)Q(S_t, a) + \alpha[r + \gamma max_{a'} Q(S_{t+1}, a')]$
In the above equation S_t refers to the previous state and S_{t+1} refers to the successive state α, γ and r are learning rate, discount factor and reward function respectively.

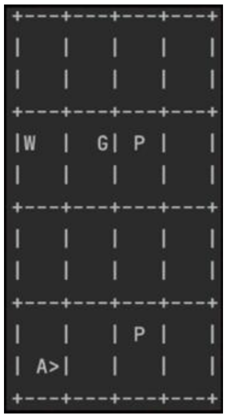
III. EVALUATION OF Q-LEARNING WITH PRECEPTS

In this section of our test run cases we performed it with the implementation of percept sensing. Astonishingly the number of possible states where the precept were included was close to 8192 possible states. The number of run iterations were kept at 6000 for each episode. Epsilon decay and alpha decay has been deployed for this approach along with greedy epsilon.

Keeping the above conditions in place, was crucial as without these boundary conditions the model would at times never converge or converge at a unreasonable time frame. But with the implementation of this condition we were able to notice the convergence rate at the 100-200 episode mark.

To explain the scoring scheme employed we notice there are times when there is a score of -6000 or -7000 by the -1 living reward accumulated over a period of time till the episode is brought to an end. Another interesting anomaly that we encountered was that if the agent does fall into the pit after an accumulated living reward of -5999, the reward value drops down to -7000.

Our implementation of an agent with percept sensing was as follows:



```
+---+---+---+---+
|   |   |   |   |
|   |   |   |   |
+---+---+---+---+
|W  |  G| P |   |
|   |   |   |   |
+---+---+---+---+
|   |   |   |   |
|   |   |   |   |
+---+---+---+---+
|   |   | P |   |
| A>|   |   |   |
+---+---+---+---+
```

Figure 1: HWA Layout

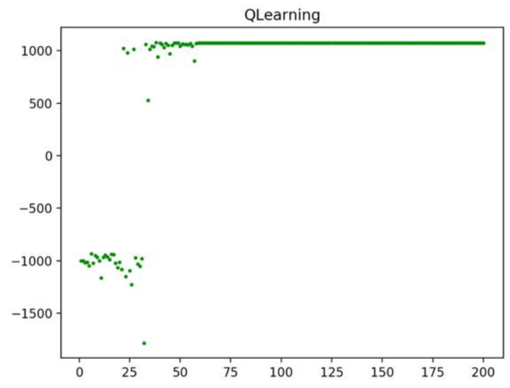


Figure 2: Without Noise Convergence Graph For HWA

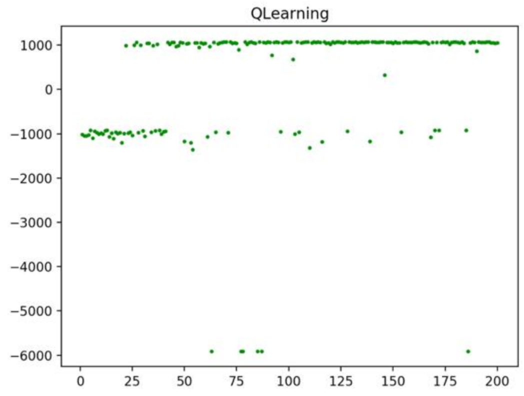
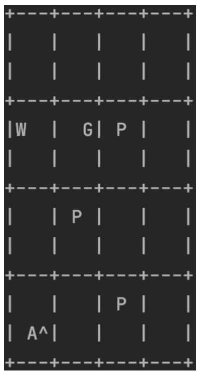


Figure 3: With Noise Convergence Graph



```
+---+---+---+---+
|   |   |   |   |
|   |   |   |   |
+---+---+---+---+
|W  |  G| P |   |
|   |   |   |   |
+---+---+---+---+
|   | P |   |   |
|   |   |   |   |
+---+---+---+---+
|   |   | P |   |
| A^|   |   |   |
+---+---+---+---+
```

Figure 4: No Option Shoot Wumpus Layout

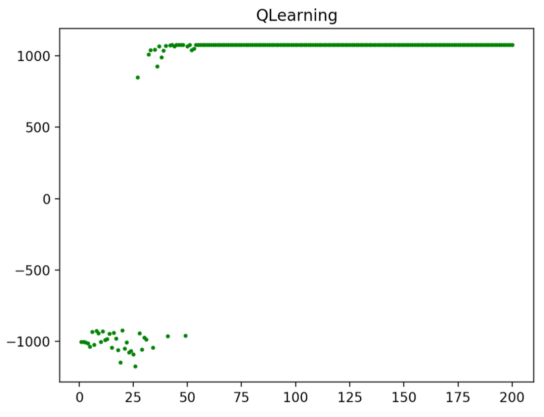


Figure 5: No Option Shoot Wumpus Without Noise Convergence

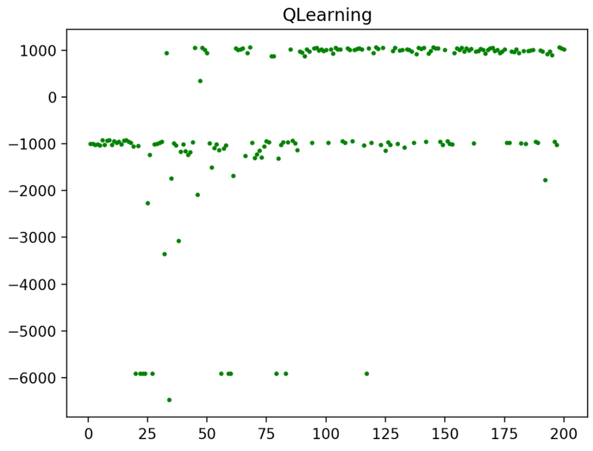
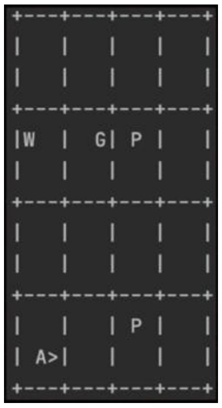


Figure 6: No Option Shoot Wumpus With Noise Convergence

IV. EVALUATION OF Q-LEARNING WITHOUT PERCEPTS

These layouts were practiced through noise and no-noise models, and further analyzed through reward v/s episode plots. In the fixed layout, it is mandatory for the agent to kill the wumpus before grabbing the gold, and this compels and proves the successful execution of the RL-agent over hybrid agent. The stochastic domain was as follows. With a random probability set at 0.2, the agent if executes forward action, then there is 0.8 probability of agent moving forward in the next time step, while a 0.1 probability of agent either slipping to the left or right. Epsilon decay has been deployed along with exploration function.



```
+---+---+---+---+
|   |   |   |   |
|   |   |   |   |
+---+---+---+---+
|W  |  G| P |   |
|   |   |   |   |
+---+---+---+---+
|   |   |   |   |
|   |   |   |   |
+---+---+---+---+
|   |   | P |   |
| A>|   |   |   |
+---+---+---+---+
```

Figure 7: HWA Layout

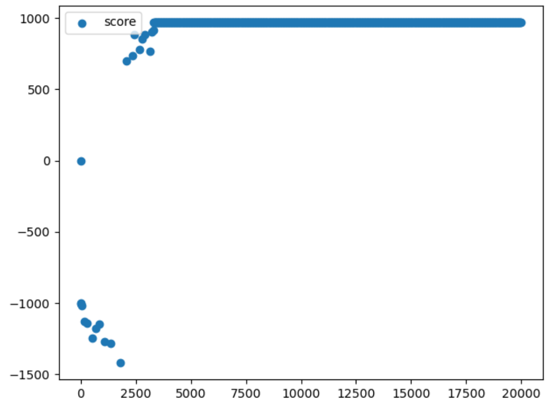


Figure 8: Without Noise Convergence Graph For HWA Without Percepts

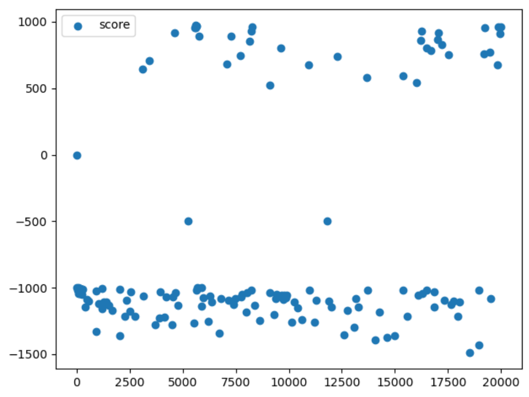
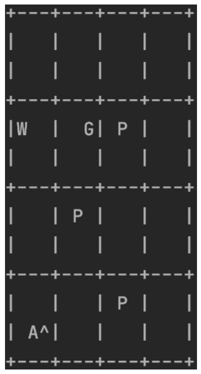


Figure 9: With Noise Convergence Graph For HWA Without Percepts



```
+---+---+---+---+
|   |   |   |   |
|   |   |   |   |
+---+---+---+---+
|W  |  G| P |   |
|   |   |   |   |
+---+---+---+---+
|   | P |   |   |
|   |   |   |   |
+---+---+---+---+
|   |   | P |   |
| A^|   |   |   |
+---+---+---+---+
```

Figure 10: No Option Shoot Wumpus Layout

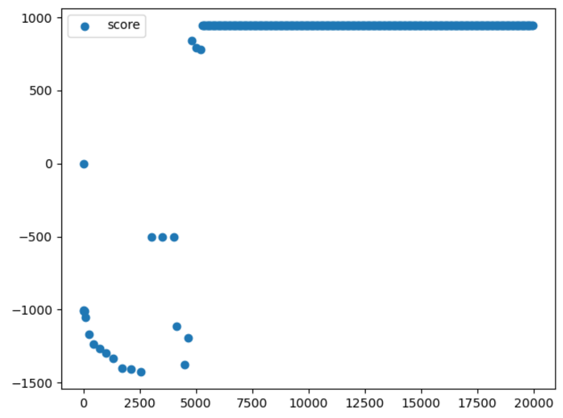


Figure 11: Without Noise Convergence Graph For No Option Layout Without Percepts

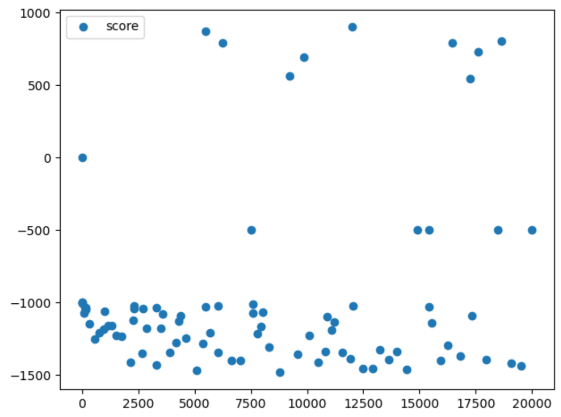


Figure 12: With Noise Convergence Graph For No Option Layout Without Percepts

V. INFERENCES/ CONCLUSION

After the agent is allowed to train we are able to observe that has a very high success rate provided it is tested upon the environment it was trained on. This performance trend is quite promising across a wide array of different environments with a very high success rate. Another interesting anomaly we did observe is that as the dimension of the environment was altered and increased the performance of the agent took a major hit.The nature of this behavior we believe is brought about because of random action take a larger time to converge with a sense of random action.

We also assumed that if the agent was exposed to a few predesigned environments and had the opportunity to train on it, it would be able to used a generalist inference for any environment but this was far from the case. We were also able to observe that the state space would become exponentially larger if the size of the environment was increased. A large state space poses two problems it first increases the memory needed increases exponentially and also increase its precept history and makes it more problematic to find the optimum policy.

But this despite its shortcomings in a few of the aforementioned areas Q-learning is significantly better in contrast to a HWA operating on a KB. It is able to operate in spaces which would normally fall outside the KB and result in the malfunction of the HWA. It also allows us the daunting task of defining a well encompassing KB which would make the HWA agent function well and saves us the task of hard-coding by having a learning factor and also there are some initial failures.

The noise simulation had also been done for our knowledge base hybrid wumpus agent (HWA) for comparison with our RL agent. The simulated noise was introduced in move forward action with 80% chance of obeying and 20% chance of executing actions either TurnLeft or TurnRight. For the 5 consistent runs agent scores -1019, -1036, -1034, -1023, -1036. In all of the runs, the HWA either fails to plan and gets stuck in a loop or takes a risk and dives into wumpus or a pit. Overall, the HWA performs poorly under the noise. In AIMA layout the HWA agent is not able to successfully solve the game where as our RL agents with both the approaches solve with fair amount of success.

The following are the video links showing the RL agent learning and also the reward vs iteration graph being plotted as the agent learns.

1) https://youtu.be/OdnHYFdTliI - Video for qLearning agent conditioned to shoot the Wumpus. Implemented with noise in agent's actions. This has 8192 possible states.

2) https://youtu.be/aewVKaE03Ts - Video for qLearning agent conditioned to shoot the Wumpus. Implemented with noise in agent's actions. This has 64 possible states.

3) https://youtu.be/vi1uqFnPsdg - Video for qLearning agent conditioned to shoot the Wumpus. Implemented without noise in agent's actions. This has 8192 possible states.

4) https://youtu.be/XrvcFcdQ0eg - Video for qLearning agent conditioned to shoot the Wumpus. Implemented without noise in agent's actions. This has 64 possible states.

VI. REFERENCES

[1] S. Russell and P. Norvig. Artificial intelligence: a modern approach. Prentice hall, Third Edition.
[2] A. Friesen. A comparison of explo- ration/exploitation techniques for a q-learning agent in the wumpus world, 2009
[3] A.Open AI wumpus gym:-https://github.com/mhenn/wumpus-gym